Fast Weather Simulation for Inverse Procedural Design of 3D Urban Models

IGNACIO GARCIA-DORADO
Purdue University and Google Research
and
DANIEL G. ALIAGA and SAIPRASANTH BHALACHANDRAN and PAUL SCHMID and DEV NIYOGI
Purdue University

We present the first realistic, physically-based, fully coupled, real-time weather design tool for use in urban procedural modeling. We merge designing of a 3D urban model with a controlled long-lasting spatiotemporal interactive simulation of weather. Starting from the fundamental dynamical equations similar to those used in state-of-the-art weather models, we present a novel simplified urban weather model for interactive graphics. Control of physically-based weather phenomena is accomplished via an inverse modeling methodology. In our results, we present several scenarios of forward design, inverse design with high-level and detailed-level weather control and optimization, as well as comparisons of our method against well-known weather simulation results and systems.

Categories and Subject Descriptors: I.3 [**Computer Graphics**]: ; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.6 [**Computer Graphics**]: Methodology and Techniques

General Terms: procedural modeling, weather, urban modeling, inverse procedural

Additional Key Words and Phrases: design, dynamical systems, inverse modeling, weather forecasting, urban climate design

ACM Reference Format:

Garcia-Dorado, I., Aliaga, D., Bhalachandran, S., Schmid, P., Niyogi, D., XXXX. Fast Weather Simulation for Inverse Procedural Design of 3D Urban Models. ACM Trans. Graph. XX, X, Article XXX (Xxxxx XXXX), yy pages.
DOI = 10.1145/1XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

Authors' addresses: I. Garcia-Dorado (corresponding author), email: ignacio.garcia.dorado@gmail.com; Daniel Aliaga, email: aliaga@purdue.edu; S. Bhalachandran, email: sbhalach@purdue.edu; P. Schmid, email: pschmid@purdue.edu; D. Niyogi, email: dniyogi@purdue.edu.

DOI 10.1145/ddddd.eeeee
http://doi.acm.org/10.1145/ddddd.eeeee

1. INTRODUCTION

In recent years, creating realistic, virtual urban environments has become an extremely important task for entertainment, education, urban planning, and training applications. This interest has fomented novel approaches to increase realism and new tools to quickly and easily design such environments. In addition to the detailed modeling of complex urban geometry, previous computer graphics work has also focused on the natural and human aspect of the city. Some works have provided methods to incorporate human behavior such as crowd simulation [Manocha and Lin 2012] and traffic simulation [Sewall et al. 2011]. Other works have focused on increasing the realism through more accurate modeling and simulation of physical phenomena (e.g., CGI movies and games use ray tracing and global illumination [Gotanda et al. 2015] and complex realistic modeling of liquids [Ando et al. 2015; Macklin and Müller 2013]).

However, relatively little attention has been paid to the design and realism of urban weather phenomena in computer graphics. Previous works have focused on the rendering aspect of weather (e.g., fog, rain, snow, clouds), but not on its generation and evolution over time. Simulating weather is difficult because of its highly non-linear behavior, sensitivity to scale-dependent phenomena, and sensitivity to initial conditions (e.g., [Pielke Sr 2013]). Typical solutions in computer graphics script the visual appearance of weather phenomena similar to key-framing in animations (e.g., tools are used to control the shape and evolution of clouds [Dobashi et al. 2008; Yuan et al. 2014; Harris and Lastra 2001] without considering the complete atmospheric conditions). This can lead to unrealistic scenarios such as clouds and rain appearing at the flip of a switch even if prior sky conditions are not indicative of rain; clouds forming, even though there is no source of vertical motion; and a mishandling of the relationship between the land surface and the clouds (i.e., urban/rural heterogeneities). Therefore, these previous solutions are useful for short highly-scripted sequences where the objective is complete control of the weather sacrificing realism and providing little automation. Further, many weather variables beyond clouds would be of interest.

Our goal is to create 3D urban models with realistic user-designed weather behavior over a long time horizon (i.e., days, weeks, even months) (Figure 1). For this, we need to create a weather model with the minimal computational cost that would allow us to i) have realistic and interactive-speed weather computation, ii) design the high-level or detailed behavior of the weather, iii) optimize weather and urban-modeling variables (e.g., for urban planning and climate adaptation studies), and iv) simulate weather indefinitely (e.g., for video games with long period in-game time passages).

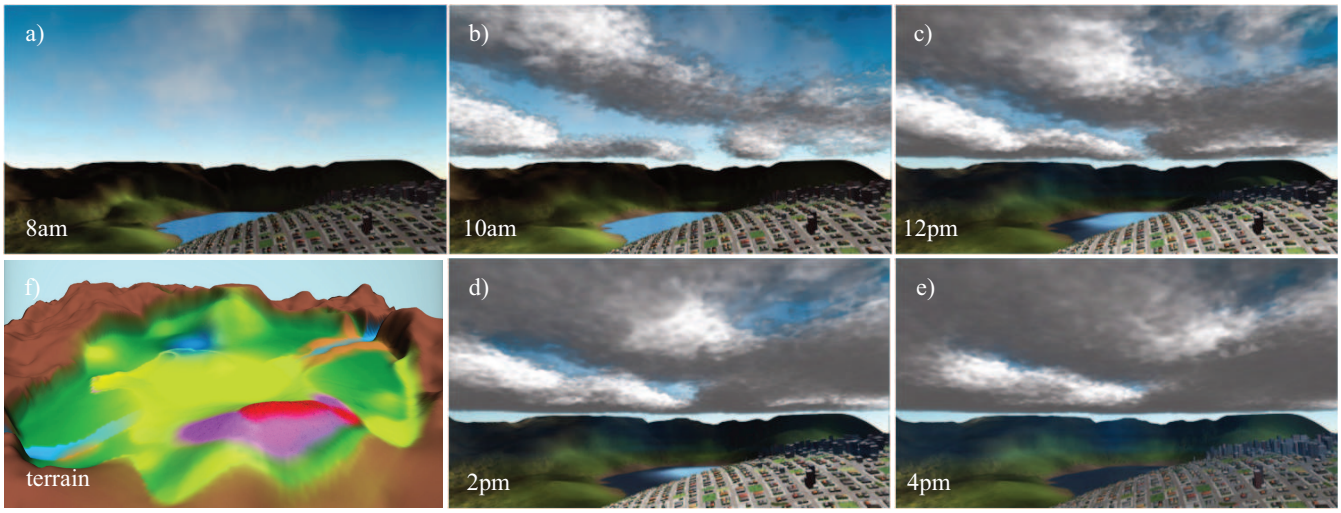

Fig. 1. **Urban Weather.** We present a method which tightly couples procedural modeling with a super real-time physically-based, interactive weather simulation. With our land use sketching interface, a user procedurally generates a terrain and 3D city (f). Then, for example, our design tools enable intuitively choosing clear sky sunrise mornings followed by afternoon convection leading to clouds and showers (a-e) without providing details about realistic spatiotemporal behavior.

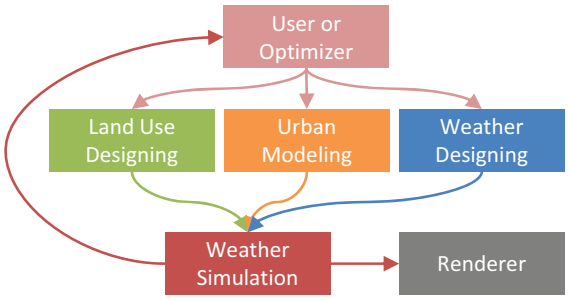

Fig. 2. **System Pipeline.** We show a diagram of the overall pipeline of our approach.

Our approach consists of three main parts: an urban procedural modeling engine (i.e., land use design and 3D urban modeling), a novel weather simulation, and an urban weather design tool (Figure 2). Altogether, our methodology is the first to enable detailed, quickly computed, and physically-based weather phenomena for procedural modeling without having to carefully script meteorological effects. Our urban procedural modeling is used to define a spatial distribution of natural and man-made land use/land cover (simply referred to as *land use* in this paper), to specify geographic location, and to indicate time and date of the year. Each land use category has a unique set of physical properties that can be computed from the generated procedural model (e.g., albedo) or obtained from empirical measurements. Our novel weather simulator provides long-lasting spatiotemporal simulations of weather based on a highly efficient fundamental equation set. Our urban weather design tool supports multiple intuitive options to simulate, alter, or design the weather pattern over time. Our tool follows an inverse design methodology to compute the set of 3D urban model parameters and initial weather parameters needed to bring the weather closer to a user-specified behavior (e.g., cloud coverage, rain, temperature). Since the solution space is large and the system non-linear, our design tool is based on a Markov Chain Monte Carlo (MCMC) optimization, similar to those used in inverse procedural modeling (e.g., [Talton et al. 2011; Vanegas et al. 2012; Garcia-Dorado et al. 2014]).

We have used our approach to simulate and render regions covering up to $2500km^2$. Note an area of this size is sufficient to cover any major urban area (e.g., in the USA they do not exceed $1600km^2$ [US-Census 2016]) and it is of a scale that is expected to affect urban weather [Schmid and Niyogi 2013]. Simulation takes from a few seconds to a few minutes of user effort to design a day long weather pattern with our application. The full 3D weather simulator computes 100 times faster than real-time (e.g., weather for an 8 hour period is computed in about 5 minutes on a desktop PC using our CUDA-based implementation). If we use our 2D *XZ* weather simulator (the same simulator as in 3D but just simulating one cross section), an approximation often done in weather forecasting and as we will show in the results with an average error smaller than 4% compared to the 3D simulator, it computes 4800 times faster than real time (e.g., the same 8-hour period is computed in 6 seconds). We present the results of weather simulations in various synthetic cities, and in comparisons to state-of-the-art weather simulators. The speed we achieved is one of its kind and has been envisaged but never achieved hitherto ([Pielke et al. 2007]).

The main contributions of our work include:

- a procedural modeling engine to sketch a terrain and city with land use categories containing physical properties for weather simulation and with plausible 3D structures,
- a novel weather/meteorological model that captures if-then impacts of urbanization and local weather, including temperature, clouds, and rain at urban neighborhood scale using a simplified, interactive, computationally fast framework, and
- a weather design tool including forward and inverse modeling methods to control the simulation.

2. RELATED WORK

2.1 Urban Procedural Modeling

Urban procedural modeling has become extremely versatile and widespread. Vanegas et al. [2010] and Musialski et al. [2013] provide surveys of urban procedural modeling and reconstruction. The seminal work of Parish and Müller [2001] and subsequent improvements (e.g., [Wonka et al. 2003; CityEngine 2016]) show impressive levels of realism. However, one well-known drawback of procedural modeling is that its inherent detail amplification makes it hard to control the output. Recently, several works have tried to overcome this limitation with inverse procedural modeling (e.g., Talton et al. [2011], Vanegas et al. [2012]). These works overcome this limitation by adding an extra layer to automatically infer the input parameters or rules to generate the desired procedural model.

2.2 Rendering Weather Phenomena

The rendering of different manifestations of weather-related phenomena has a long history in graphics. For example, simple fog has become a part of the OpenGL specification, Blinn [1982] presented a light reflection method to simulate clouds, and virtual snow has been shown (e.g., Nishita et al. [1997]). More recently, rendering work has focused on further improving photorealism, such as Garg and Nayar [2006] who concentrate on rain streaks. Cloud rendering has received much attention as well. Harris and Lastra [2001] use a preprocessing step for simulating scattering and employ impostors at runtime. Bouthors et al. [2008] present an algorithm for realistic real-time simulation of multiple anisotropic scattering in clouds. They propose a new formulation for the macroscopic behavior of light requiring just to transverse the boundaries of the cloud. Yuan et al. [2014] present a method for estimating the shape of a symmetrical cloud from a single image by inverting a single scattering model. Other works have centered on the simulation of weathered appearance in urban environments. For instance, Chen et al. [2005] present a visual simulation technique that uses aging-inducing particles to simulate a wide variety of weathering phenomena. Bosch et al. [2011] extract the effect of fluid flow from captured images to generate synthetic weathering effects. However, all these methods focus only on producing a compelling 3D rendering of a weather-related phenomena and not on simulating and predicting coupled, dynamical weather over a significant time period.

2.3 Cloud Simulation

The most closely-related simulations to our work in computer graphics are various forms of cloud simulation over time. Kajiya and Von Herzen [1984] present an early method to solve the scattering equations and present equations which model the dynamic behavior of clouds. Dobashi et al. [2000] present computationally inexpensive cellular automata to model cloud evolution using several simple transition rules and offline rendered clouds. Miyazaki et al. [2001] use a coupled map lattice to approximate the formation of various cloud shapes. Overby et al. [2002] modify a fluid solver to simulate clouds based on buoyancy, relative humidity, and condensation. Harris et al. [2003] describe a cloud simulation engine. However, their system does not have a prognostic simulation of cloud variables, radiative transfer model, rain variables, nor procedurally generated models of land use. Dobashi et al. [2008] enable a user to draw the contour of a cumuliform cloud from a specific camera position and their system automatically adjusts parameters so that the simulated result fits in the drawn contour. They assume constant heat from the ground and do not take into account wind, radiation, and surface energy balance. These systems have not developed an environment that creates and balances the energy from the surface.

In contrast to and building upon the above methods, our approach is based on a full weather simulation model that includes surface energy balance, boundary layer processes, and second order evolution of meteorological forcing equations ([Holton and Hakim 2012]). Also, most prior approaches are diagnostic, i.e., weather is a static input that does not change, and the objective is to get the visualization of cloud and other phenomena which are mutually affecting each other. This interdependency becomes essential for scenarios where a priori specification of clouds or environment is not an option. Examples include serious game environments, battlefield theater scenes, or education and synthesis modules where the visualization is used as part of a decision system. In each of these examples, the decision either by the model or the user is intimately tied to the graphics that emerge as a result of the environmental and initial conditions. Providing a suitable solution for the graphics community is one of the goals of this paper, as well as provide very fast simulation performance.

2.4 Weather Forecast Models

Clouds and precipitation events are one of the most difficult features to accurately simulate in a prognostic weather model. This is because their simulation and prediction require all other factors to be adequately represented and simulated, and errors in any of the simulated variables can translate into inaccurately developing clouds or numeral instability. Nebeker [1995] and Stull [2000] provide a comprehensive review and history of Numerical Weather Prediction (NWP). Weather forecasting can be done at a variety of scales ranging from global-scale simulation (e.g., jet streams) to micro-scale simulation (e.g., wind effects between buildings). For our goal, we seek a physically-based, interactive-speed urban-scale simulation that models weather over and around a city and a range of land uses. The urban simulation capability we pursue is not present in the majority of simplified weather simulation systems – only complex state-of-the-art weather research is addressing our desired urban feedbacks and it typically requires a large amount of computational and storage resources such as those available on supercomputers and clusters.

Implementations of several well-known weather forecast models are available. The Weather Research and Forecasting (WRF) Model [Skamarock et al. 2005; Chen et al. 2011] is one of the state-of-the-art NWP systems designed to serve both atmospheric research and local operational forecasting needs. The Regional Atmospheric Modeling System (RAMS) [Cotton et al. 2003] is another such NWP model and a mesoscale weather simulator (e.g., for horizontal scales of up to $2000km$ and grid cell sizes of up to $20km$) for weather and climate research. Though in our tests and configurations, WRF is faster than RAMS, both systems are extremely computationally expensive. For example, a 72-hour WRF simulation for a $50x50km$ area and $25km$ high (using 50x50x56 grid cells) takes around ten hours using one compute server of 48 cores.

We base our approach on the general methodology of WRF (and provide additional comparisons in Section 7.3). Attempts have been published to accelerate WRF but nothing near the level of performance we pursue (e.g., on the order of minutes). Michalakes and Vachharajani [2008] use GPUs to accelerate a portion of the WRF model resulting in only an overall 1.23x faster system. Mielikainen et al. [Mielikainen et al. 2013] use CUDA to accelerate a single thread implementation of WRF's cloud and precipitation module

by 70x; even after mapping their relative improvement to our hardware and usage of WRF, we obtain a further 10x to 490x speedup of a *full* weather implementation (i.e., not just the cloud and precipitation module).

Note that our goal is not to develop a faster WRF; instead, our objective is to extract a subset of model equations and parameterization schemes, and to perform simplifications, suitable for simulating urban-scale weather phenomena over and near a city, as well as create a subset of parameters that are essential for our analysis to be processed and rendered. We discuss WRF mainly because of its popularity and because we will be using it for comparisons.

3. OVERVIEW

Our method enables a user to interactively sketch an initial urban and non-urban area, simulate or design the weather over a desired time frame, and output the user-specified behavior of weather suitable for rendering. Figure 2 and this section provide an outline.

3.1 Procedural Modeling and Land Use

Using an interactive tool, the user *paints* a distribution of land use categories (e.g., location of urban area, location of agriculture, forests, water bodies) over the terrain surface, the terrain can be optionally altered, and geometry is procedurally generated. Once land use is defined, a link with the weather model is achieved through the provision of initial conditions and surface physics parameters (e.g., albedo). The spatial distribution of land use categories is used as input to our procedural modeling engine (Figure 3). As in Parish and Müeller [2001] and CityEngine, roads, city-blocks, and parcels are generated procedurally. In our current implementation, urban input parameters control the procedural generation: urban uses (origin of the road generation, main road axis, area of influence), roads (irregularity, number of lanes, and intersections), blocks and parcels (area, percentage of parks), and buildings (building type, stories, setbacks, distribution of white roofs, ratio of window-to-wall). Additionally, urban vegetation and urban amenities are produced.

Our method supports the following twelve categories of urban and non-urban scenarios (based on typical usage in weather modeling):

- bare ground,
- desert/beach sand,
- high mountains,
- grass (e.g., grass fields, prairies),
- forest (e.g., tree and dense/tall vegetation)
- snow, (e.g., typically on higher-elevation mountains)
- water (e.g., river, lake),
- crops (e.g., corn, wheat),
- low-density residential (e.g., houses),
- high-density residential (e.g., apartment buildings),
- low-density industrial/commercial, and
- high-density industrial/commercial (e.g., factories).

The physical properties of each land use category are computed using other variables and using the procedural geometry (see Supplemental Material B). The land use variables consist of: surface albedo, soil heat capacity per unit area C_{GA}, ground layer thickness d_s, the Bowen ratio B_o, mean seasonal surface temperature T_m, average building height, and surface roughness. These variables will be discussed in more detail in Section 4. Note that the twelve land use categories being considered can be easily updated and increased to a larger (or lesser) number as needed.

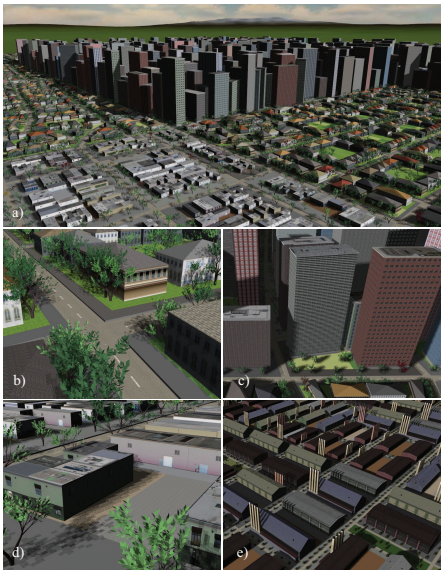

Fig. 3. **Urban Procedural Modeling.** Our method uses a) procedural modeling to generate a city and terrain, including b) low-density residential, c) high-density residential, d) low-density industrial, and e) high-density industrial.

3.2 Weather Model

Our realistic urban weather model is fast enough to be useful for graphics and for urban planning scenarios but also accurate enough to simulate regional weather. As presented in related works, current realistic weather models are very complex and are not designed for real time, interactive, vis a vis change in landscape configurations and related applications [Chen and Dudhia 2001; Jantz et al. 2010]. Our novel GPU based weather model performs preprocessing, simulation, and post-processing with increased performance mainly because:

- We initialize the simulation with input from a real-world or synthetic atmospheric sounding, periodic boundary conditions, and user-defined land use. As a result, format conversions and data integration is significantly simplified.
- Our fundamental equations during the simulation are non-hydrostatic and comprise of the advection term, diffusion term and the buoyancy term and include simplifications specific to the range of grid spacing pertinent to our interest (about 1x1km). In particular, we focus on local weather supporting only a single resolution grid (as opposed to the nested-grids provided in other systems to account for multi-scale and external phenomena). This avoids the huge overhead of re-sampling and transforming variables between different grid resolutions and different models. Moreover, it does not include subgrid resolution phenomena nor account for forcings assuming large grid sizes (e.g., the Coriolis Effect, which does not have a very significant impact at our scale of interest, is not included).
- The radiation and energy balance component during simulation is typically relatively expensive to compute; thus we carefully designed a simple force-restore based method with empirical simplifications that works very effectively.
- We use a scale-specific differentiation scheme that is simple yet robust and fast.

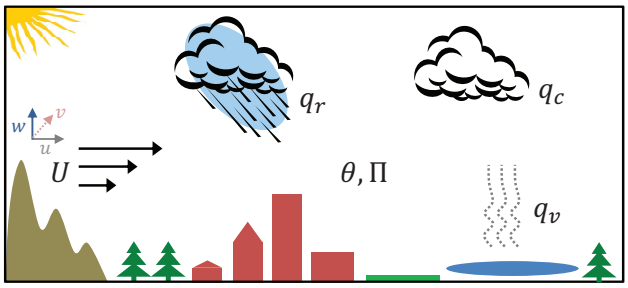

Fig. 4. **Weather Variables.** We show a depiction of the weather variables computed by our simulator.

- Finally, based on a thorough and critical analysis we identified the most time consuming aspects and optimized them for better GPU performance.

From the visual perspective, the physical weather-related phenomena we want to particularly consider are wind, humidity, clouds, and rain. Including these phenomena in a realistic weather simulator implies creating a system that includes: i) fundamental equations, ii) a radiation and energy balance model, and iii) a cumulus cloud microphysics model. Our optimized fundamental equations (Section 4.1) capture boundary layer physics and mass energy exchanges and dynamics. Further, since various physical phenomena start with the sun heating the surface, we include a radiation model (Section 4.3) that simulates solar radiation, its conversion into heat, and the initiation of weather changes via surface heterogeneities. Moreover, we include a microphysics model (Section 4.2) that governs the bigger conversions between water vapor, liquid cloud water, and liquid rain. Finally, we integrate this with the simplest differentiation scheme that is stable at the scale of our grid size and time steps, and place it within a GPU friendly architecture.

3.2.1 *Variables and Grids*

Our approach uses the following variables stored within a 3D grid structure over the simulated region (Figure 4):

- **Wind velocity** ($U = \{u, v, w\}$): These are the 3D wind components (computed in m/s) in the west-east, north-south, and vertical direction, respectively, and provide both the mean as well as turbulent fluctuating components.
- **Potential temperature** (θ): The potential temperature of a grid cell of air is the final temperature the cell would have attained if it were moved adiabatically (i.e., without heat or mass transfer) from pressure p and temperature T (in Kelvin) to a standard pressure $p_{z=0} = 100kPa$ (i.e., approximately sea level pressure). Its definition can be written as

$$\theta = \frac{T}{\Pi} \tag{1}$$

- **Exner function** (Π): The Exner function is a non-dimensional pressure quantity designed to simplify pressure-related computations in atmospheric modeling. It is written as

$$\Pi = \left(\frac{p}{p_{z=0}}\right)^{R_d/c_{pd}} \tag{2}$$

where R_d is the specific gas constant of dry air ($R_d = 287.058J/(kg \cdot K)$) and c_{pd} is the heat capacity of dry air at constant pressure ($c_{pd} = 1004.5J/(kg \cdot K)$).

- **Moisture variables** (q_v, q_c, q_r): These are mixing ratios of water vapor, cloud water, and rain water (non-dimensional quantities of mass-of-air per mass-of-air).

Our method uses the standard Arakawa C-Grid [Arakawa and Lamb 1977], commonly used in the weather community, to offset mass and energy variables in vertical and horizontal directions. A C-Grid is preferred for weather models because it prevents physically unrealistic waves of non-wave variables (i.e., temperature) from forming in the model, and performs better for second order differentiation [Purser and Leslie 1988]. In such a grid, the scalar variables are defined in the center of each grid cell, and the vector variables are prescribed in the faces. In the horizontal plane, the ground is split into evenly spaced grid cells typically of 1 kilometer tiles. In the vertical direction, the grid spacing is log-linear with higher resolution closer to the surface *boundary layer* (i.e., the first few kilometers of the atmosphere which are most influential to weather processes of our interest) and then gradually increasing in size (i.e., becoming coarser). This is also consistent with the vertical gradients observed in the atmosphere with typically higher values closer to the surface, and higher layers showing relatively well mixed atmosphere. Thus, the user defines the first grid cell height d_z and the rest is calculated following the power law

$$z[k] = min(d_z \cdot s_r^k, d_{max}) \tag{3}$$

where d_z is typically $50m$, stretching ratio s_r is usually 1.025, and the maximum height d_{max} is $1000m$. This approximation is based on the Monin-Obukhov Similarity Theory profile estimation [Monin and Obukhov 1954]. Thus, each vertical column of cells has 12 grid cells under $3km$ (i.e., in the region that is expected to be influenced by the boundary layer) and the topmost grid cell reaches $25km$.

The weather variables are initialized using a *sounding*. A sounding is an observed or analyzed atmospheric profile from the real-world of temperature, moisture, and wind and used for prescribing the initial states of variables within weather forecasting. These soundings can be obtained from an external source, from gridded reanalyses [Mesinger et al. 2006], through a single sounding that can be interpolated to each grid location based on surface characteristics, or can be generated procedurally (see Section 5.1). The lateral boundaries of the grid are assumed periodic (i.e., toroidal in 3D). The bottom boundary conditions are determined from the land surface and energy balance model, and the top boundary conditions are described in Section 4.5.

3.2.2 *Dynamical Equations*

As mentioned, our nonlinear dynamical equations consist of three components:

- *fundamental modeling:* accounts for advection, diffusion, and buoyancy of wind, humidity, temperature, and Exner function.
- *cloud and precipitation modeling:* accounts for evaporation, condensation, auto-conversion, accretion of water, and potential temperature.
- *radiation modeling and land use:* accounts for short-wave and long-wave radiative flux onto the ground and urban surfaces, and the resulting temperature and humidity changes.

In our system, these three components are the minimum set of equations needed to create a physically-based, dynamic simulation that models temperature, wind, clouds, and rain.

One option, as presented in Dobashi et al. [2008], to yield weather phenomena would be to force the changes synthetically

(e.g., instant changes of temperature in a grid-cell without a physical reason). However, this leads to unrealistic situations in a coupled mode and also to an increased requirement of scripting weather phenomena particularly when larger time horizons are desired (e.g., days, weeks, months). Instead, our physically based weather simulation improves both automaticity and realism in computer graphics. As an example, a more urban land cover against a vegetation cover would conservatively be warmer by 1-3 degrees depending on the radiation, energy and surface characteristics. Moreover, since the temperature changes depend on the type of land use, our systems allows quickly obtaining different realistic weather behaviors by just altering land use. Without the cloud and precipitation model, there would not be clouds nor rain. Further, although the temperature, pressure, and wind might change, the effect on weather would not be very noticeable. Our fundamental model equations are crucial to tie these components together. Thus, each of the model components feeds into and build off the other model components. Indeed, this is the nature of the weather processes seen in reality as well.

3.3 Design Tool

Our design tool provides several options to create a desired weather pattern. Initial conditions define the environment where the simulation begins. Our framework enables two main levels of design: forward design that allows unlimited weather simulation for urban procedural models, and inverse design to control the weather. This inverse design allows users to control weather from a high-level perspective, achieve specific user weather behaviors, or control the weather in a visual sense.

4. URBAN WEATHER SIMULATION

Given a set of initial values either calculated or provided, our weather simulator updates grid cell variables during each time step by integrating finite difference partial differential equations described in this section.

4.1 Fundamental Component

Our set of fundamental or primitive equations is derived from the laws of motion and thermodynamics and refined using scale-analysis for our desired urban-scale weather simulation. We take the well-known basic equations that model momentum, mass continuity, and energy conservation [Holton and Hakim 2012], and define a new subset model based on those equations that are as simple as possible to reduce computational expenses yet still yield relevant weather phenomena. This is done using domain knowledge: simplifying terms when possible, refactoring terms using scale analysis and constants that are reasonable and choosing the exact form to work efficiently in a parallel implementation. An example of this would be that the Coriolis effect (i.e., a force produced by the rotation of the Earth) is removed from the equations since our scale is only on the order of tens of kilometers – we would need a several order of magnitude bigger scale to notice a numerical change in the simulation values.

The equations model the motion of the wind (U), temperature via potential temperature (θ), and atmospheric pressure and density via the Exner function (Π). The equations simulate advection, diffusion, mass and energy balance, and buoyancy [Holton and Hakim 2012]. Advection (e.g., $U \cdot \nabla$) is the transport mechanism of a substance by a fluid due to the mean flow motion (as against turbulent transport). For instance, liquid water that forms the cloud droplets is advected by wind causing the clouds to move in the sky. Diffusion (e.g., $\kappa\nabla^2$) represents molecular and turbulent exchange of a substance from a region of high concentration to a region of low concentration. For example, warm air spreads to the surrounding colder areas. For a parcel of air in hydrostatic balance, there is a balance between gravity and the pressure gradient force and hence air does not move vertically. However, if there is a change in density (e.g., in ideal gases due to a change in temperature or pressure), a buoyant force is exerted. If the parcel is less dense (than the surroundings), the parcel exerts an upward force; if it is denser, then the net force is downwards. This fundamental interlinking of terms and processes is central to the model functioning (simplest to most state of the art ones). As a verification test, we will show that a cold bubble sinks and a warm parcel of air rises in our results (Section 7.3).

To improve numerical stability in our implementation and to simplify some formulas to save computational cost, each variable is decomposed into its base value and a time-varying perturbation value (e.g., $\phi = \phi_{t=0} + \phi'$ where ϕ is the instantaneous value of a vector or scalar variable, ϕ_0 is the base value, and ϕ' is the perturbation as a result of external forces). To illustrate the conversion of the equations to actual code, we have added an example of a complete equation with its discretization in Supplemental Material A. For notation brevity, we use $\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + (U \cdot \nabla)$, called the material derivative. Moreover, we denote time steps as: current n, former $n-1$, next $n+1$, and $t = 0$ as the initial value.

Wind. The equations modeling the change of the horizontal wind velocities u and v are

$$\frac{Du}{Dt} = -c_{pd}\theta_v \frac{\partial \Pi'}{\partial x} + \kappa\nabla^2 u_{n-1}, \text{ and} \tag{4}$$

$$\frac{Dv}{Dt} = -c_{pd}\theta_v \frac{\partial \Pi'}{\partial y} + \kappa\nabla^2 v_{n-1}. \tag{5}$$

The right-hand side of Equations (4) and (5) include two terms:

- The first term represents acceleration due to the pressure gradient force expressed by the Exner function and θ_v is the density potential temperature which accounts for both liquid and vapor water in air density ($\theta_v = \theta_{t=0}(1.0 + 0.61 q_{v_{t=0}})$).
- The second term represents wind diffusion based on the winds in the previous time step and by the diffusion coefficient in each component κ (typically, $\kappa = (500, 500, 100)$).

The equation for modeling the change of the w wind velocity (i.e., vertical wind) is

$$\frac{Dw}{Dt} = -c_{pd}\theta_v \frac{\partial \Pi'}{\partial z} + g\left(\frac{\theta'}{\theta_{t=0}} + 0.61 \cdot q'_v - q'_c - q'_r\right) + \kappa\nabla^2 w_{n-1}, \tag{6}$$

where the right-hand side includes a first and third term similar to the u and v wind velocities. The second term is the computation of vertical buoyancy as a potential temperature of a parcel (perturbation) compared to its surroundings (mean state) with moisture added to represent acceleration from falling liquid water. The vertical velocity term controls much of the convection processes and the vertical exchanges in the boundary layer.

Pressure. The Exner function value is updated using the following prognostic equation [Durran 1989]

$$\frac{D\Pi'}{Dt} = \frac{c_s^2}{c_{pd}\rho\theta_v^2}\nabla \cdot (\rho\theta_\rho U) + \kappa\nabla^2 \Pi'_{n-1}, \tag{7}$$

where $c_s = 100m/s$ is about one-third the speed of sound (a simplification by Durran to obtain an anelastic approximation considering the temperature dependence of the speed of sound) and ρ is

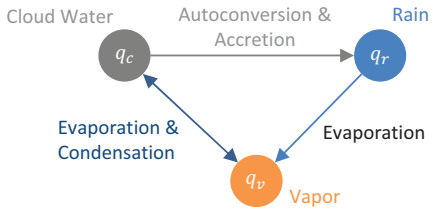

Fig. 5. **Kessler Microphysics.** Water transition states within the model clouds following the Kesller scheme.

air density (in kg/m^3). The first term computes advection and the second term computes pressure diffusion.

Temperature. The potential temperature perturbation values are updated using

$$\frac{D\theta'}{Dt} = -\rho w \frac{\partial \theta'}{\partial z} + \kappa \nabla^2 \theta'_{n-1}. \tag{8}$$

The first term represents the change in temperature due to non-adiabatic vertical advection and the second term represents its diffusion.

4.2 Clouds and Precipitation Component

To simulate clouds and precipitation at urban scale, we build off the conceptual approach suggested in Pielke et al. [2007], but we focus on convection and warm rain processes. Further, we use the classic equations derived from the Kessler scheme [Kessler 1969] and the mesoscale framework of Soong and Ogura [1973] to adapt this conceptual approach to our 3D grid version with our set of fundamental equations and variables.

4.2.1 *Microphysics Scheme*

The Kessler scheme is a warm (liquid-only) cloud scheme that includes water vapor, cloud water, and rain. The equations use a variety of constants determined experimentally. In this scheme, cloud formation and dissipation are estimated through condensation and evaporation, rain drop growth via the growth of cloud droplets (autoconversion), and the collection of droplets from falling rain (accretion). Figure 5 depicts the model framework.

Condensation. This is the process by which water vapor in the air is changed into liquid water and is implicitly linked to cloud formation and precipitation. The expression for condensation from vapor (q_v) to liquid (q_c) is implemented based on the simplification by Tetens [1930] as

$$C_{v\mapsto c} = \min\left\{q_v, \frac{q_v - q_{vs}}{1.0 + q_{vs}\frac{a(273-36)L_{lv}/c_{pd}}{(\Pi\theta)^2}}\right\} \tag{9}$$

where $a = 7.5ln(10)$ and $L_{lv} = 2.501 \cdot 10^6 Jkg^{-1}$ corresponds to the latent heat of vaporization. The use of the minimum operator ensures mass balance between water vapor (q_v) and liquid water (q_c). The equation also makes use of the theoretical maximum amount of water vapor that air at a specific temperature and pressure can hold. This is known as the saturation mixing ratio and is implemented in our model as

$$q_{vs} = b \exp\left(a\frac{\Pi\theta - 273}{\Pi\theta - 36}\right) \tag{10}$$

where $b = 380.16/p_e$, and $p_e = p_{z=0}\Pi^{c_{pd}/R_d}$.

Evaporation. This can be considered as an inverse process of condensation: liquid water (q_c and q_r) turns into water vapor (q_v). The model considers the air saturation potential and when the air becomes unsaturated, cloud droplets (q_c) evaporate to maintain a balance between liquid water in the atmosphere and water vapor. Thus, the rate of evaporation is self-regulated so as to keep the air at the saturation mixing ratio until the cloud droplets are completely evaporated. Raindrops (q_r) evaporate after cloud droplets are condensed by the moisture deficit. Evaporation is calculated as

$$E_{c,r\mapsto v} = \min\begin{cases} q_r \\ -C_{v\mapsto c} - q_c \\ V_c\frac{q_{vs}-q_v}{\rho q_{vs}}\frac{(\rho q_r)^{0.525}}{5.4\cdot 10^5 + 2.55\cdot 10^8/(\bar{p}q_{vs})} \end{cases}. \tag{11}$$

The top term ensures evaporation does not exceed the rain, the middle term verifies the newly available vapor is not exceeded, and the third empirically determined term ensures the rate to maintain the saturation mixing ratio. The third term also uses the ventilation coefficient V_c to consider air pollution concentration near the ground surface (particularly relevant in the context of an urban scenario). It is calculated by the experimentally-determined equation

$$V_c = 1.6 + 124.9\rho q_c^{0.2046}. \tag{12}$$

Autoconversion. This process computes rain droplets to be formed if cloud water exceeds a critical value. This is calculated as

$$A_{c\mapsto r} = \begin{cases} 0.0 & \text{if } q_v \le q_{c0} \\ \max\left(0.0, 10^{-3}(q'_c - q_{c0})\right) & \text{if } q_v > q_{c0} \end{cases} \tag{13}$$

where $q_{c_0} = 10^{-3} g/kg$.

Accretion. This last process represents when rain collects the small cloud droplets while falling. Avoiding snow and ice computations in this implementation, accretion can be approximated by

$$B_{c\mapsto r} = \max\left(0.0, 2.2\, q'_c q'^{0.875}_r\right). \tag{14}$$

4.2.2 *Model Implementation*

Finally, we put together the previously described microphysics concepts to write a set of per-grid cell water equations, representing clouds and precipitation, which conserve total net mass:

$$\frac{Dq_v}{Dt} = -\rho w\frac{\partial q_v}{\partial z} + \kappa\nabla^2 q_v - C_{v\mapsto c} + E_{c,r\mapsto v}, \tag{15}$$

$$\frac{Dq_c}{Dt} = \kappa\nabla^2 q_c + C_{v\mapsto c} - A_{c\mapsto r} - B_{c\mapsto r}, \tag{16}$$

$$\frac{Dq_r}{Dt} = \kappa\nabla^2 q_r + \frac{1}{\rho}\frac{\partial \rho V_t q_r}{\partial z} + A_{c\mapsto r} + B_{c\mapsto r} - E_{c,r\mapsto v}, \tag{17}$$

where $V_t = 36.34\sqrt{\frac{\rho_{z=0}}{\rho}}(\rho q_r)^{0.1364}$ is the rain water terminal velocity and $\rho_{z=0}$ is the density of the lowest grid cell per column. These equations also make use of the material derivative (as in Section 4.1). Considering the energy change associated with water phase change (i.e., evaporative cooling, condensation heating), the thermodynamic equation is updated (in lieu of Equation (8)) as

$$\frac{D\theta'}{Dt} = -\rho w\frac{\partial \theta'}{\partial z} + \kappa\nabla^2\theta'_{n-1} + \frac{L_{lv}}{c_{pd}\Pi}(C_{v\mapsto c} - E_{c,r\mapsto v}), \tag{18}$$

where on the right-hand side the first term and second term are the same as in Equation (8) and the third term is the non-adiabatic heating (cooling) due to phase change.

4.3 Radiative Energy Flux and Energy Balance Component

The third set of equations models how solar radiative flux is estimated, partitioned, and interacts with the surface and atmosphere causing a change in temperature that initiates all the changes in the weather dynamics. Our radiation model considers the solar radiation and its interaction with the surface and atmosphere by using a force-restore slab method, based on the one described by Stull [1988]. We improve this method with three additions: i) ray marching to compute cloud coverage, ii) extending the simulation time by removing the DC variations and biases of the model – the original model was designed for only 24 hour time simulations, iii) modeling the temperature exchange of this model with the bottommost grid of the system by achieving coupling.

In particular, the objective of our radiation model is to compute a spatially and temporally changing value for the temperature of each of the bottommost grid cells – we refer to such temperatures as T_z. The model defines constants, variables, and equations that are effectively below the bottommost grid and depend directly on the distribution of land use. Each constant of each bottommost grid cell is computed at the beginning of the simulation as a linear combination of land use types (these constants can be found at Supplemental Material B).

Once T_z is updated, the weather dynamics start with the exchange of temperature that produces buoyancy, and the energy, mass, dynamics-based prognostic equations. Note that in the land use interface (i.e., transition region between two land use types of adjacent grid cells) is where buoyancy occurs and where the exchange of temperature and water starts. These processes, in turn, define where clouds will more likely form. The exchanges are modeled following Monin-Obukhov Similarity Theory [Monin and Obukhov 1954].

4.3.1 *Radiation budget*

The radiation flux corresponding to one ground-level grid cell is split into four parts of a two-stream model,

$$Q^* = K\uparrow + K\downarrow + I\uparrow + I\downarrow, \tag{19}$$

where the shortwave solar radiation is reflected ($K\uparrow$) and transmitted ($K\downarrow$) and the longwave radiation is emitted ($I\uparrow$) up and diffusively radiated down ($I\downarrow$).

Shortwave radiation. The visible light transmitted by the sun can be quantized by

$$K\downarrow = S_c T_K \sin(\Psi) \tag{20}$$

where the solar constant $S_c = -1.127K\,m/s$ is the intensity of incoming solar radiation at the top of the atmosphere, and T_K is the radiation attenuated by the depth of atmosphere it has to travel (e.g., at sunset the radiation has a longer path to reach the surface) and by the amount of clouds, aerosols, and other absorption/reflection components within the atmospheric layer.

To compute T_K, the model discretizes cloud coverage into three different heights: low-level cumulus, mid-level alto, and high-level stratus (i.e., typically low 0-2km, medium 2-6km, and high $>$ 6km clouds); with the fraction at each height being σ_{C_L}, σ_{C_M}, and σ_{C_H}, respectively.

Stull [1988] uses an approximation to define cloud coverage. In our system, to compute the amount of clouds for each bottom-level grid cell, we use a 3D ray marching method which sets the ray origin as the center of each grid cell, the ray direction as the vector to the sun, and a ray step size adequate to sample each cell vertically. This sampling is a coarse approximation especially for angles close to the horizon; however, in those cases the contribution of the sun to heat the surface are negligible and the error is very small. Then, we accumulate the result of applying a cloud transfer function (see Section 6) to each of the ray marching samples within each of the three height ranges, yielding σ_{C_L}, σ_{C_M}, and σ_{C_H}. The contribution (or weight) by the amount of clouds at each of the three heights, and other weights, can be dynamically altered or empirically set such as the simplifications introduced in Stull [1988]. Hence,

$$T_K = (0.6+0.5\sin(\Psi))(1-0.4\sigma_{C_L})(1-0.7\sigma_{C_M})(1-0.4\sigma_{C_H}). \tag{21}$$

Note that the values 0.4, 0.7 etc. used for cloud coverage are merely examples and can be changed by the user to influence the cloud formation at the lower, mid and high altitudes.

The solar elevation angle Ψ is determined by the longitude/latitude of the urban space, time of day, day, and year:

$$\sin(\Psi) = \sin(g_{lat})\sin(d_s) - \cos(g_{lat})\cos(\delta_s) \cos[(\pi t_{UTC})/12 - g_{long}] \tag{22}$$

where g_{lat} and g_{long} are geographic latitude and longitude of the middle of the simulated region, t_{UTC} is the time in UTC, and δ_s is the solar declination angle:

$$\delta_s = 0.409\cos\left(\frac{2\pi(d-173)}{365.25}\right) \tag{23}$$

where d is the day of the simulated year (and 173 represents the summer solstice day and 365.25 the number of days in a year).

The reflected shortwave radiation $K\uparrow$ is the fraction of $K\downarrow$ that is reflected by the surface as defined by its albedo (a):

$$K\uparrow = -aK\downarrow. \tag{24}$$

Longwave radiation. Earth re-emits the solar energy as radiation in the form of longwave infrared rays. The net longwave radiation, $I^* = I\uparrow + I\downarrow$, is modeled by the empirically determined equation [Stull 1988]:

$$I^* = 0.08(1 - 0.1\sigma_{C_H} - 0.3\sigma_{C_M} - 0.1\sigma_{C_L}). \tag{25}$$

4.3.2 *Force-Restore Slab Model*

Figure 6 pictorially represents the multiple temperature layers and the model variables of the force-restore method [Blackadar 1978; Chen and Dudhia 2001]. As stated, the surface energy balance is performed by splitting the radiation (Q^*) into sensible heat flux (Q_H), latent heat flux (Q_L), and ground heat flux (Q_G). The division of surface energy defines four layers in the radiation model:

- Two atmospheric layers consist of (*a*) the bottom of the model grid with temperature T_z and (*b*) a layer near the surface with temperature T_a.
- Two ground layers consist of (*c*) a shallow layer of d_s centimeter thick and temperature T_g where most of the soil temperature changes occur because of the sun's radiation and thermal diffusivity and (*d*) a deeper layer primarily influenced by seasonally varying mean temperature T_m.

To compute the temperature T_z at height z, we start using a diurnal empiric *environmental lapse rate* value γ and the temperature of air closest to the surface, T_a. The empirical formulation of γ is represented by

$$\gamma = 1.07\cdot10^{-8}t_l^5 + 8.18\cdot10^{-7}t_l^4 - 6.05\cdot10^{-5}t_l^3 + 7.72\cdot10^{-4}t_l^2 + 1.4\cdot10^{-3}t_l - 0.0184. \tag{26}$$

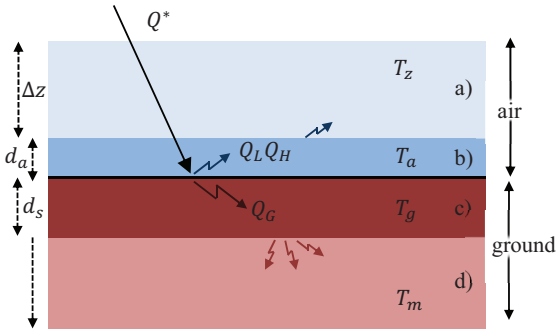

Fig. 6. **Force-Restore Slab Model.** Radiation reaches the surface and this heats T_g; a fraction gets diffused to deeper layers of the ground T_m, and a part heats the first few cm of air T_a, then T_a heats the bottom layer of model grid T_z.

where t_l is the local time. Thus, T_z is computed as

$$T_z(z) = T_a + \gamma z. \tag{27}$$

In order to compute T_a, we first compute the ground heat flux

$$-Q_G = C_{GA}\frac{\partial T_g}{\partial t} + 2\pi\frac{C_{GA}}{P}(T_g - T_m) \tag{28}$$

where C_{GA} is the soil heat capacity per unit area. Supplemental Material B contains a table with values for a, C_{GA}, d_s, and B_o. These values can be found in standard textbooks [Stull 1988; Noilhan and Planton 1989; Oke 2002].

To compute the change of T_g, we make use of the aforementioned radiation budget Q^* and the heat fluxes out of its layer interfaces as

$$\frac{\partial T_g}{\partial t} = \frac{-Q^*}{C_{GA}} + \frac{2\pi}{P}(T_m - T_g) - a_{FR}(T_g - T_a) - T_{gc}. \tag{29}$$

On the right-hand side, the first term is the force term quantifying the amount of radiation that reaches the surfaces and is transformed into heat depending on the soil heat capacity C_{GA}; the second term is the restoration term measuring the conduction of temperature T_g to the deeper layer of temperature T_m; the third term represents thermal convection of the ground layer of temperature T_g to the first layer of air with temperature T_a, where a_{FR} is the conductivity between the ground and the air ($a_{FR} = 3 \cdot 10^{-4}$ when $T_g > T_a$ and $a_{FR} = 1.1 \cdot 10^{-4}$ when $T_g \leq T_a$). Note that this formulation is not balanced within a 24 hour period, i.e., there is a net temperature increase over a one day time period. As is typical in weather models, we add a correction factor, T_{g_C}, to remove this DC component and it is computed as $T_{g_C} = [T_g(t = 86400) - T_g(t = 0)]\,\Delta t/(86400)$, i.e, the fraction of net increase in each time step during the first 24 hour period. In a similar manner, we correct T_a with T_{a_C}.

Having computed the radiation Q^* and the ground heat flux Q_G, the sensible heat flux is computed by radiative balance. We extend the Blackadar [1978] model to compute Q_H using the Bowen ratio B_o (i.e., ratio of sensible heat flux to latent heat flux):

$$Q_H = (-Q^* + Q_G)\frac{1}{1 + B_o}. \tag{30}$$

Finally, we estimate the change in T_a using a temporal function of sensible heat flux. Assuming heating from the surface is the sole source of increasing or decreasing air temperature and a constant boundary layer height ($z_i \approx 1km$), we compute it as:

$$\frac{\partial T_a}{\partial t} = z_i^{-1} Q_H - T_{aC}. \tag{31}$$

Note that the first iteration of the Equation 29 requires the value of T_a. We use the bottommost grid cell of the atmospheric sounding to initialize its value.

4.4 Numerical Stability

To improve the numerical stability and computational robustness during the simulation, we make several optimizations for time and space integration.

For space integration, we use second-order central differentiation with the aforementioned Arakawa C-Grid. For time integration, we use leapfrog integration and a Robert-Asselin time filter [Durran 2013].

Leapfrog numerical techniques define the next time step of each variable as

$$\phi_{n+1} = \phi_{n-1} + 2\Delta t\frac{\partial \phi_n}{\partial t}, \tag{32}$$

where Δt is the time step. Leapfrog, instead of using the current time step value to compute the next, uses the former time step value and duplicates the Δt to reach (i.e., leap) to the next time step. This temporal offset of the integrated quantities enables achieving second order accuracy thus increasing the stability and enabling larger time steps which leads to a faster runtime.

In our implementation, we include the well-known Robert-Asselin time filter to further increase the stability. We compute each variable as

$$\phi_{n+1} = \bar{\phi}_{n-1} + 2\Delta t\frac{\partial \phi_n}{\partial t}, \tag{33}$$

$$\bar{\phi}_n = \phi_n + \lambda(\phi_{n+1} - 2\phi_n + \phi_{f_{n-1}}). \tag{34}$$

where λ is a filter factor with values $\lambda \in (0.06, 0.35)$ (which represents the filter range from global to advection-diffusion problems). In our system, we use $\lambda = 0.1$. Note that for the first iteration, ϕ_f is not available; thus, we use the standard forward in time equation $\phi_{n+1} = \phi_n + \Delta t\frac{\partial \phi}{\partial t}$.

To analyze the numerical stability, we use the Courant-Friedrichs-Lewy (CFL) criteria. From Durran [2013], leapfrog and Robert-Asselin have a CFL criteria of $s = 1$. This means that in order to converge $\Delta t\,(u_{max}/\Delta x + v_{max}/\Delta y + w_{max}/\Delta z) \leq 1$. Given that $u_{max}, v_{max} < 50m/s$ in our simulation (i.e., higher winds are only found in tornadoes) and $w_{max} < 30m/s$, the maximum time step for $\Delta x = \Delta y = 1000m$ and $\Delta z \geq 200m$ is $\Delta t \leq 4$ seconds. Thus, our model satisfies this necessary condition using a $\Delta t \leq 4$ seconds. Note that CFL is a necessary but not a sufficient condition of stability, there are other types of numerical instability that can arise. However, CFL usually gets violated first so it is a good marker of when integration will become stable. In practice, the combination of leapfrog, Robert-Asselin, C-Grid, and integrating the momentum terms in flux form allows our system to be stable using a $\Delta t = 1$ second (the value used for all experiments).

4.5 Boundary Conditions

For top and bottom boundaries, we set zero wind and moisture variables. Potential temperature, Exner function, and pressure are de-

fined as the boundary value stretched out through the vertical domain:

$$\begin{cases} \phi(z<0) = \phi(z \geq grid_z) = 0 & \text{for } \phi = \{u, v, w, q_*\} \\ \phi(z<0) = \phi(z=0) & \text{for } \phi = \{\theta, \Pi, \rho\} \\ \phi(z \geq grid_z) = \phi(z = (grid_z - 1)) & \text{for } \phi = \{\theta, \Pi, \rho\}. \end{cases} \tag{35}$$

Note that the lower boundary is set by the radiation and energy balance model.

4.6 Stochastic Subgrid Model

Cloud formation, especially cumulus convection, is a multiscale feature with features ranging from turbulence and small-scale vortices and whirls to large-scale subsidence due to differential mesoscale convergence/divergence fields. Therefore, even at the relatively fine-resolution of this model (e.g., $1km$ grid spacing) considered in our study (compared to the $10km$ used in operational numerical weather models), we need to consider the features impacting cloud evolution from the sub-grid scales. The smallest feature that a given grid scale Δx can resolve would be one that is $\sim 4\Delta x$. Atmospheric turbulence occurs at scales much smaller than the finest grid spacing tested by the model ($\ll 1m$) and contributes to some of the variability of cloud development and the resulting heterogeneity witnessed in the real atmosphere. To represent the visual effects of subgrid-scale variability, we implement two different approaches.

First, we follow an approach similar to Randall and Huffman [1980] by splitting the model equations into a deterministic component (ϕ), and a stochastic component (ϕ_s). The deterministic component is the physically derived differential equations of our system. The stochastic component represents the physically significant but unresolvable features of the subgrid scale. For this purpose, we add small random variations to the potential temperature of the surface grid cells and to the q_c of the forming clouds (i.e., grid cells with $0.001 < q_c < 0.002$). We define $\phi_s = \mathcal{N}(0, 0.1\,\phi)$, i.e., up to 10% variability. This perturbation is gradually added to each time step over a 5 minute period. The stochasticity in the potential temperature at the surface grid cells was added in the radiation and energy balance model to simulate the heterogeneities in the land surface and at the cloud centers was added to the moisture variables to simulate the impact of aerosols. When integrated over time and space, these stochastic variables are expected to average out to zero and thus have an effect just within the cloud.

In addition to the aforementioned technique, to reintroduce the lost rotational motion, we use the vorticity confinement approach defined by Steinhoff and Underhill [1994] and later implemented for graphics applications by Fedkiw et al. [2001]. The methodology introduces an additional force term perpendicular to the gradient vorticity field, thereby increasing the local vorticity.

5. URBAN WEATHER DESIGN TOOL

Our tool provides either forward or inverse design options for city models and local weather. As mentioned, in the forward design, the user interactively draws the land use of urban and non-urban regions and defines the initial weather conditions. The local weather is then simulated in a prognostic manner. In inverse design, our system discovers how to change the land use distribution or the initial weather conditions to cause the weather/climate pattern to behave as desired. Uses of these design options are shown in the results section (Section 7).

5.1 Input Parameters

Our system has three sets of initial input parameter values: $\Omega = \{\omega_l, \omega_p, \omega_w\}$.

- The ω_l parameters refer to the percentage distribution of land use for each grid cell. For example, in the center of the city there is a nearly 100% likelihood of urban land use. This distribution can be defined with our interactive drawing tool, be loaded from a GIS data (e.g., WRF global fields) or model databases, or be procedurally generated.
- The ω_p parameters refer to the urban procedural parameters (Section 3.1) that define the urban geometry (e.g., building height mean, road width, percentage of white roofs).
- The ω_w parameters refer to the initial conditions of the weather simulation. These conditions define the initial values for each grid cell for each simulation weather variable. These conditions can be defined explicitly, procedurally, or via observations. Our system generates soundings procedurally or uses a small database of measured soundings (e.g., the public online database provided by the University of Wyoming, Department of Atmospheric Science and NOAA's National Centers for Environmenta Information (NCEI), provide worldwide data for the last several decades, example is in Supplemental Material C). To generate the initial conditions procedurally, we define spline control points at vertical heights of $\{0, 2, 6, 8\}km$ and fit a spline to interpolate values for each variable at each height. In our system, the spline values can be defined from physically-based range values (e.g., $u \in (0, 70)m/s$) or from a predefined list (e.g., cold/warm/hot, dry/humid).

Note all these parameters can be spatiotemporally varying. For instance, the sounding can vary by either prescribing it from a weather model output and interpolation to each grid cell, or by using terrain and surface layer similarity/mixed layer similarity approximations [Arya 1999] to re-estimate the sounding. In our simulations, we set the values initially and run the simulation.

5.2 Forward Design

In forward design, the user wants to quickly design a city model and simulate realistic weather for different time periods. This approach is useful to virtual environments, modern games, education, and decision tools where the goal is realism by taking advantage of the detailed amplification of procedural modeling. The user defines directly the desired input parameters. Usually, this means drawing the land use distribution (ω_l) and choosing some initial weather conditions (ω_w). Given these initial conditions, our weather simulator is then executed for the desired time period. Note that the simulation varies from one to the next day because i) the initial conditions are altered through the former day, ii) the radiation model that directly affect the wind and buoyancy changes with the day of the year due to the sun trajectory, and iii) the potential of rain also changes depending on the moisture, cloudiness and seasonality.

5.3 Inverse Design

Building upon the inverse procedural modeling concept (e.g., Bokeloh et al. [2010], Talton et al. [2011], Vanegas et al. [2012]), we also provide a novel inverse modeling tool for designing urban weather (e.g., control cloud coverage, humidity, rain distribution, or city temperature). Given a procedurally-generated model, we discover how to alter the model or the initial conditions so as to produce a user-specified weather. Since weather simulation is a very

nonlinear and complex process, it is very hard to predict and/or control the end state. Therefore, we propose an MCMC-based method, in particular, based on the Metropolis-Hasting algorithm [Metropolis et al. 1953; Hastings 1970], to explore the search space and find a solution that exhibits the desired weather behavior.

In our method, the optimization mode for inverse design may be one of the following:

- *Error (objective) function optimization.* This mode minimizes a function $E(*)$ that describes the desired behavior. The function can be as simple as a single value of a weather variable at the end of the simulation (e.g., temperature in a specific grid cell) to a set of more complex values (e.g., precipitation over different periods of times, percentage of sunlight in a grid cell during a day).
- *Cost minimization.* This mode allows to achieve a desired behavior while minimizing the number of changes. The user defines an objective value η for a function $E(*)$ and defines a cost function $C(*)$ to minimize. For example, in urban planning a user might want to control a weather variable while doing the minimum changes to the urban model.
- *Constrained optimization.* This defines constraints to the solution parameters, and it is an option for the other two modes. For example, the user wants to constrain the solution to be no more than say 10% different than the original.

5.3.1 *Seeding Initialization*

Our method starts from one or more initial parameter values or *seeds* Ω_0. The user can choose whether to use a single seed consisting of the parameter values of the original model or multiple seeds that consist of physically-based valid parameter sets. In practice, a useful configuration to control the weather without changing the 3D model is to fix $\{\omega_l, \omega_p\}$ and alter the initial weather conditions ω_w. In this case, we create multiple seeds with different ω_w parameter values. Using domain knowledge in weather simulation, we know that significantly varying temperature/clouds/rain can be produced by altering the gradients in landscape and the initial wind (i.e., U) and humidity (i.e., q_v) values at multiple heights. To speed up the convergence and focus on physically possible behaviors, we precompute a wide range of physically valid values for ω_w (128 in our configuration) in different land use distributions (10 distributions in our configuration). These values are then used to automatically select a discrete set of initial seeds close to the objective and, thus, likely to achieve fast convergence. Moreover, we use these precomputed values to evaluate the solution feasibility (Section 5.4). Note that without this seeding, the optimization would require more random initial seeds or need more steps to find the desired solution. We also have the means to store and reuse the preconditions already plotted or computed to be used for future cases.

5.3.2 *Search Process*

Subsequently, our method attempts n_s state changes $\Omega_t \rightarrow \Omega_{t+1}$ starting at the seeds and attempts to better satisfy the user-specified behavior. The search is performed using one or more search threads each having a different energy level within an MCMC optimization framework. Given the current state Ω_t, a candidate next state is computed by changing one or more values of the three parameters sets.

For parameter values in ω_l and ω_p, the user selects a set of grid cells where the changes can happen (or the whole scenario), the allowable changes of land use (e.g., change from forest to low-density residential and green), and subset of procedural parameters that can change (e.g., only the window-to-wall ratio and roof albedo can change). To calculate the next state Ω_{t+1}, our system randomly selects a subset of the permissible grid cells and performs a random perturbation to the land use or a procedural parameter in each grid cell. For each parameter value ω and its physically-possible range $(\omega_{min}, \omega_{max})$, we perturb the current value with a value sampled from a Gaussian distribution $\mathcal{N}(0, \alpha|\omega_{max} - \omega_{min}|)$ where α is the perturbation change (typically set to a random value $\alpha \in (0 - 10\%)$). For constrained optimization mode, we clamp the final value to enforce the constraints. If the change increases (or decreases) a land use type in a grid cell, the other land use types in the same grid cell are randomly decreased (or increased) by the same amount so that the sum of all land use distributions stays at 100%.

For parameter values in ω_w, the user selects the weather variables that can be altered and the plausible physical range (or the default physically-based range values are used). Then, using the same sampling scheme, the variables are perturbed.

5.3.3 *Acceptance Ratio*

Once a candidate state Ω_{t+1} has been sampled, we re-simulate the weather and use a modified Metropolis ratio to compute the probability to accept this new candidate state. Note that if the candidate state Ω_{t+1} is not accepted, the transition $\Omega_t \rightarrow \Omega_{t+1}$ does not occur and the next state is Ω_t again.

A standard Metropolis-Hasting acceptance ratio when the proposed sampling distribution is symmetric is defined as

$$a_{min}(\Omega_t \rightarrow \Omega_{t+1}) = \min\left\{e^{-\beta[E(\Omega_{t+1}) - E(\Omega_t)]}, 1.0\right\}. \quad (36)$$

However, our acceptance ratio is computed as

$$a_{min}(\Omega_t \rightarrow \Omega_{t+1}) = \begin{cases} \min\left\{e^{-\beta\left(\frac{E(\Omega_{t+1})}{E(\Omega_t)} - 1.0\right)}, 1.0\right\} & \text{if } E(\Omega_{t+1}) \geq \eta \\ \min\left\{e^{-\beta\left(\frac{C(\Omega_{t+1})}{C(\Omega_t)} - 1.0\right)}, 1.0\right\} & \text{if } E(\Omega_{t+1}) < \eta \end{cases} \quad (37)$$

where β is the energy level that affects the acceptance ratio, $E(*)$ is an error (objective) function, and $C(*)$ is a cost function. If β is small (~ 1.0), then it is more likely that higher error candidate states are accepted. In contrast, when β is larger, it is more likely to accept only improved candidate states.

Our formulation differs from the standard one in two significant aspects:

- Our acceptance ratio is based on relative improvement. For the standard acceptance ratio of Equation 36 to work, the error function $E(*)$ should produce similar values during the entire search process, or normalization factors should be computed and used. However, computing normalization factors is challenging when the error function varies by several orders of magnitude during the optimization iterations or when the range of values is difficult to define a priori. Instead, our approach eliminates the need to find adequate normalization factors by creating an acceptance ratio that only depends on the relative improvement. To accomplish this, our formulation replaces the difference computation of the error functions by a division of them. In Figure 7, we show the behavior of the acceptance ratio. The actual value of $E(*)$ is irrelevant, and only the relative error is important and independent of the energy level.
- Our modified formulation also serves to implement the error optimization and cost minimization modes (Section 5.3). In error

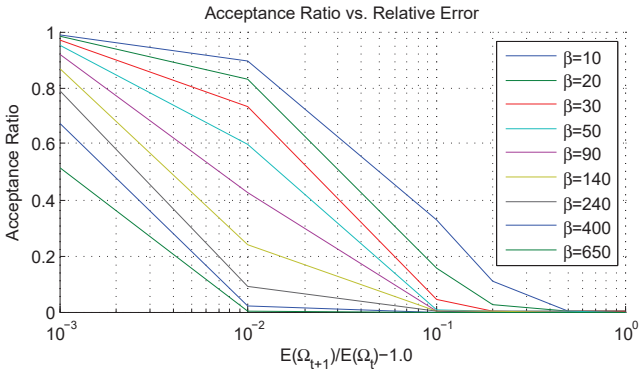

Fig. 7. **Acceptance Ratio vs. Relative Error.** The figure shows the acceptance ratio for different energy levels. While the energy level (β) decreases, the acceptance ratio decreases as well, making more unlikely that worse states get accepted.

optimization mode, the function is simply optimized as in standard MCMC. This is accomplished by setting $\eta = -\infty$ (i.e., the threshold is effectively deactivated). In our cost minimization mode, two behaviors will alternate. While the $E(*)$ is not satisfied (i.e., $E(*) \geq \eta$), the optimization will minimize $E(*)$. In contrast, when the objective value is achieved (i.e., $E(*) < \eta$), the optimization minimizes a second function $C(*)$ (i.e., the cost function). Using this modification, we can optimize a variable but with the minimum number of changes (or cost). Figure 12 shown ahead is an example – once the temperature objective is reached, the optimization then tries to reduce cost alternating between both behaviors, and the search stays close to the $E(*) \approx \eta$. See Section 7.2 for more details.

5.4 Incoming Weather Design

In addition to our novel weather simulation, we also provide an *Incoming Weather Design* mode. In this mode, we remove the toroidal boundary conditions and, instead, feed in external (i.e., incoming) user-defined states. This mode is useful in two cases: a) to overcome the limitations of a local simulation, and b) allow the user to define specific cloud shapes over different periods of time.

A limitation of a local-only simulation is that not all user-defined weather behaviors are possible – some local simulation behaviors require a particular global scenario (e.g., an overcast day in desert area is not possible unless there is an incoming/global overcast behavior carrying water from elsewhere).

Hence, our design tool enables the user to pick (or define) the weather state outside of the local simulation region (from precomputed or user-defined states) and thus effectively control the global weather scenario. The user selects several external states and defines a timeline for each of them. These states are then fed in as boundary conditions changing over time. Since in reality, weather at a location is due to features advected from another locale, we include this functionality by forcing the wind to advect inwards across the boundary, where we can 'receive' the selected states as an incoming source, thus collectively yielding the desired weather behavior. Moreover, the user can decide whether the local weather should interact with the incoming one. This is done by altering the initial soundings. For example, using a sounding with a high water vapor mixing ratio and/or different wind speeds will cause interaction between local and incoming advected and moisture values.

Note that this method can also be used even if the weather behavior is achievable. Using the incoming weather, the user can define a concrete set of clouds and have a more tight control of its spatio-temporal behavior.

6. IMPLEMENTATION DETAILS

Our system is implemented in C/C++ and CUDA and renders using OpenGL. For the CPU simulation, we use C++ Boost threads (for multi-threading) and Boost mutex (for synchronization).

While not the focus of our work, we provide real-time rendering using three render passes. In a first pass, the sky is rendered using a GPU version of Nishita et. al. [1996] where the sky's appearance is a combination of Mie and Rayleigh scattering. In the same pass, we optionally add a lens flare effect and parametrized 4D procedural noise for far background clouds. In a second pass, the terrain and urban procedural model is rendered. To improve performance, we reduce the number of draw calls by grouping together all geometry with the same texture. Shadows are added based on current cloud coverage (Section 4.3.1) and a shadow mapping technique enhanced with a Poisson Disk (Dunbar et al. [2006]). In the final pass, clouds and rain are rendered using 3D ray marching and volumetric rendering based on Wrenninge and Bin Zafar [2011]. We enhance this approach with the idea of Volume Perturbation as presented by Elbert [2003] and Kniss et. al. [2003]. Our method defines an empirically determined cloud transfer function and rain transfer function based on q_c and q_r values respectively. Weather variables are visualized in real-time using 2D or 3D textures, an isoline shader, and/or ray marching.

For the GPU implementation, each component is implemented as a single CUDA kernel so that the block size can be optimized independently. To improve SIMD performance, we store grid variables as a structure of arrays (rather than an array of structures) to coalesce reads and writes. To reduce the impact of branching, variable access is implemented via an efficient array operator overloading scheme.

7. RESULTS

We have implemented and tested our model, the framework, and the rendering approach to design and simulate weather on various cities. All computation is performed on an Intel i7 4770 CPU and an NVIDIA GeForce GTX 780 graphics card.

7.1 Forward Design

Figure 8 shows two exemplar cases of the method described in Section 5.2. The user draws the distribution of land use (Figure 8a) and designs the procedural model of the city (Figure 8b). Then, the user can select initial procedural weather conditions (top row) or select a location and date and load real-world soundings (bottom row), in this case from Miami, FL. Finally, our system simulates any number of days of realistic weather (Figure 8c).

7.2 Inverse Design

We demonstrate our weather design tool and related assessments in Figures 9, 11, and 12. We show how our framework can be used to design, control, and optimize different scenarios.

Cloud Design.

Figure 9 shows three examples of the use of our high-level design tool. The user wants to control the cloud coverage (e.g., percentage of sky covered by clouds) of a procedural city throughout a day. To calculate the cloud coverage, we compute the average amount

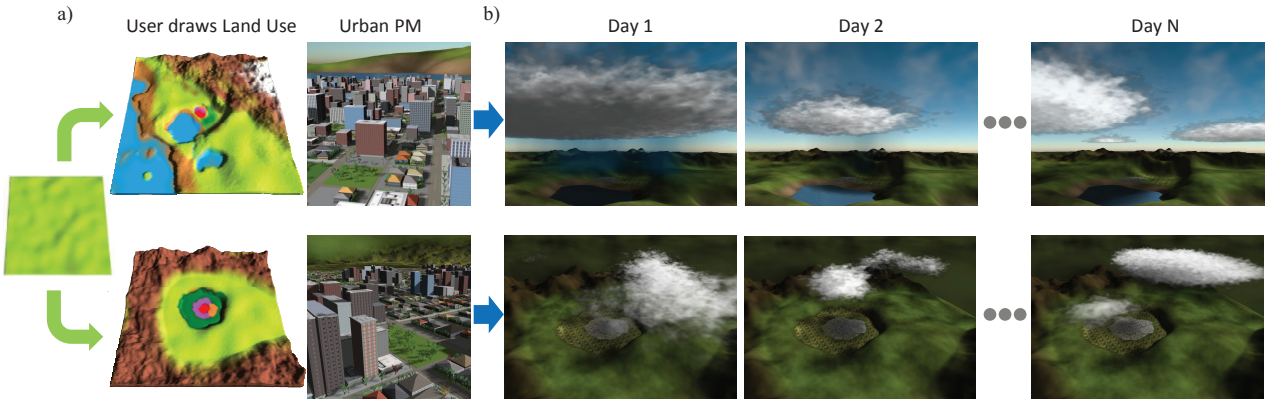

Fig. 8. **Forward Design.** Two examples of forward design. a) The user draws the land use distribution and uses our urban procedural model engine to design two cities; b) the simulation runs for these procedural models and we display different days.

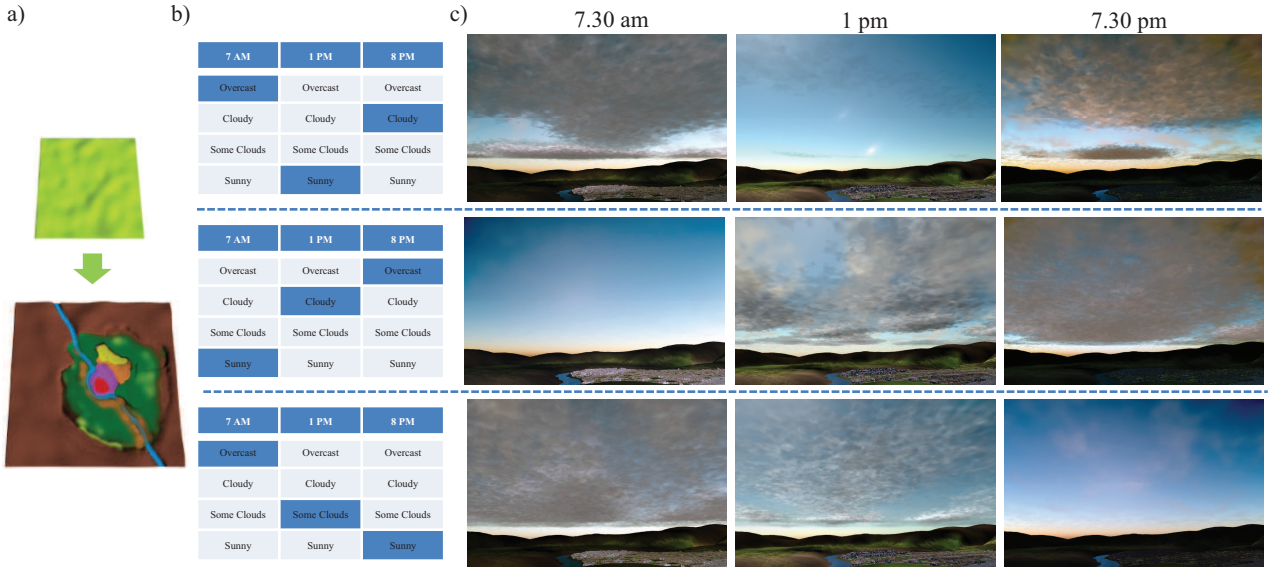

Fig. 9. **Inverse Cloud Design.** Three examples of cloud design. a) The user interactively draws a land use distribution; b) the user selects three different high-level behaviors of the weather; c) the system finds such weather and the weather sequence is visualized for different days.

of clouds computed by our Radiation and Energy balance model (Section 4.3.1).

The process is as follows. The user interactively paints an urban and non-urban area (Figure 9a). Then, the user designs the weather behavior (Figure 9b) by defining as an objective the mean percentage of cloud coverage ($\bar{\sigma}$) and, optionally, the permissible range around the mean ($\check{\sigma}$) during m different time ranges R. This defines an objective cloud coverage set $\{\bar{\sigma}_{r\in R}, \check{\sigma}_{r\in R}\}$ where $R = \{r_1, r_2, \cdots, r_m\}$. Then, we run the optimization to find the closest solution (Figure 9c).

For this example, we fix ω_l and ω_p since we do not want to alter the 3D model. To alter ω_w, we set-up the optimization i) to use our multi-seed approach, ii) to constrain the variable values within plausible physical values, iii) to only alter the control points of U and q_v, iv) and to use $\eta = -\infty$ (i.e., error optimization mode). Finally, we define the error function to minimize the computed cloud coverage with respect to the user-specified one as

$$E(\Omega) = \sum_{i=1}^{m} \max\left(0.0, |\sigma_{\Omega r\in R} - \bar{\sigma}_{r\in R}| - \check{\sigma}_{r\in R}\right) \qquad (38)$$

To speed up the process and for certain applications, the user can decide to use our 2D *XZ* simulation model. Our 2D *XZ* model uses the same formulation and code than for 3D but drops one-dimension (in this case the depth, *Y*) to create a fast approximation for cloud coverage. Figure 10 shows the error for three different scenarios of the computed cloud coverage using our complete 3D version and our 2D *XZ* approximation. As can be observed the error for time periods of 12 hours have an average value smaller than 4% and a maximum error of 10%. We use this 2D approximation in this result, but in general it could be used in many other time-critical applications.

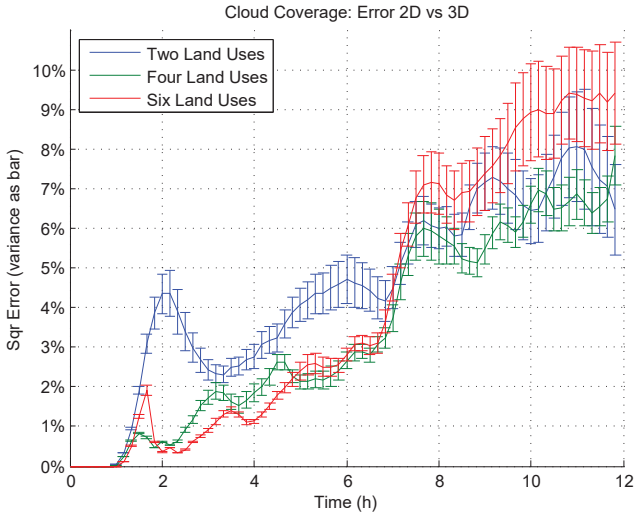

Fig. 10. **Difference between 2D and 3D simulations.** We show the squared error of the cloud coverage between our 2D and 3D simulators, for three different scenarios and with 50 different initial conditions.

The presented solutions were found using 12 initial seeds, 4 steps, and required less than 5 minutes of computation. Optionally, the user can decide to explore the rest of evaluated solutions (plotted as temporal cloud coverage changes) and select a more suitable simulation.

Rain Design.

Figure 11 presents an example of our inverse modeling tool to control (e.g., decrease or increase) the rain of the city by changing non-urban land use. Altering the rain rates of an area can be interesting for many reasons. An example would be to alleviate seasonal droughts or flood potential ([Lucke and Nichols 2015]). Decreasing rain may be desired for humid and rainy areas (e.g., Seattle, WA) or even to mitigate the spread of insects such as mosquitoes. A wide range of environmental services have been envisaged for urban greening, and an optimization tool such as discussed here is critically required.

In this example, we focus on changing non-urban land use since it is relatively cheaper as well as more feasible than re-designing an existing urban area. Nonetheless, in developing countries, as well as design changes planned over long time horizons, planning the future development of the urban landscape itself is crucial. Regardless, such land-use change exploration is being actively studied ([Zhang et al. 2009; Gero et al. 2006]). To limit further the solution, we define the maximum percentage of land use change of each grid-cell to be 50%.

For this example, we set ω_p to the default procedural parameters and ω_w with values corresponding to a warm summer day. To control the rain, we alter ω_l of the non-urban areas (green in Figure 11c) allowing changes to 'bare soil', 'forest', 'beach sand', and 'crops'. We use our system with one unique seed (i.e, the original scenario) and run it with 20 chains of different energy levels $\beta \in [10, 650]$ and 50 steps. In Figure 11a-b we present the evolution of rain for each chain in gray and highlight as a thick line the minimum/maximum value found so far for that number of steps. The black circled state is the one selected as optimal.

To compute the total rain fallen in h hours, we use the concept of *rain intensity*. Each s seconds of simulation we sample the rain intensity (in mm/h) in each bottom level grid-cell. We

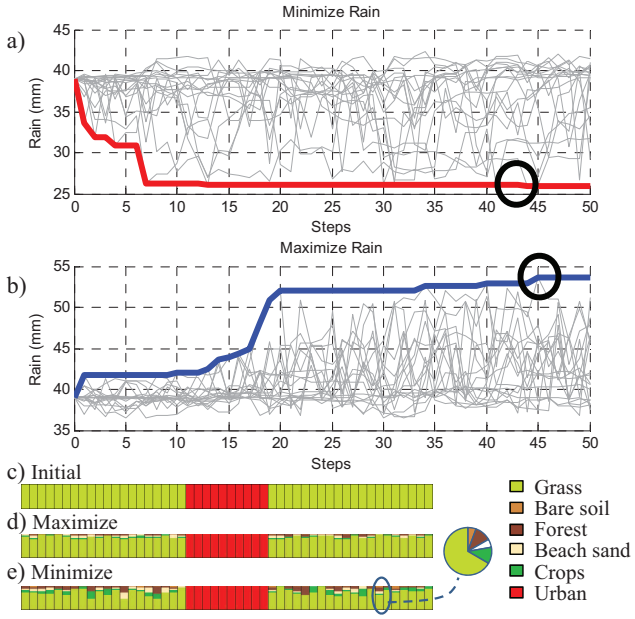

Fig. 11. **Inverse Rain Design.** We show the results to optimize the rain levels in a city, a) to decrease it; and b) to increase it as a function of urban landcover only. We overlap the result of each chain in gray and highlight the minimum/maximum value found so far; c) shows the initial land use distribution of the center cross section of the urban region; d-e) displays the best solution for each optimization corresponding to the black circles on a and b).

use the classic [Marshall and Palmer 1948] formula to compute the rain intensity as $RI = 360\rho_{z=0}V_r q_r$ where terminal velocity is $V_r = 3634(\rho q_v)^{0.1354}$. The total rain is computed as the average of *RI* per hour times the elapsed time (in our case 12 hours); i.e.,

$$E(\Omega_t) = \sum_{t=0}^{h} \frac{3600}{s} \sum_{i=0}^{grid_x} RI(t,i). \tag{39}$$

Using this formula as the objective function, rain is minimized (Figure 11a). In order to use the same function but as maximization (Figure 11b), we invert the terms in the acceptance ratio to be $E(\Omega_t)/E(\Omega_{t+1})$ instead of $E(\Omega_{t+1})/E(\Omega_t)$. As seen in the particular case of Figure 11, we achieve a rain decrease of 45% and an increase of 35%. Note that such 'drastic' changes are indeed possible since we included 'beach sand' as a land use category. This land use category has a very high albedo. In contrast, if we eliminate this option, the maximum decrease is only 19% and the maximum increase is 16%. These outcomes have both educational if-then visualization value, and also of relevance to the consideration of interactive realism within meteorological analysis.

Temperature Design.

Figure 12 presents an example of our cost minimization mode. We use our system to reduce 1° C of temperature in the city center (i.e., $\eta = [E(\Omega_0) - 1]$ Celsius) so as to alleviate the higher temperatures produced by the urban heat island effect. Reducing the urban heat island by only a few degrees is recognized as a difficult though valuable goal [Solecki et al. 2005]. For example, super dense cities such as Beijing can benefit from reduced heat islands that would in turn decrease pollution [Mullaney et al. 2015].

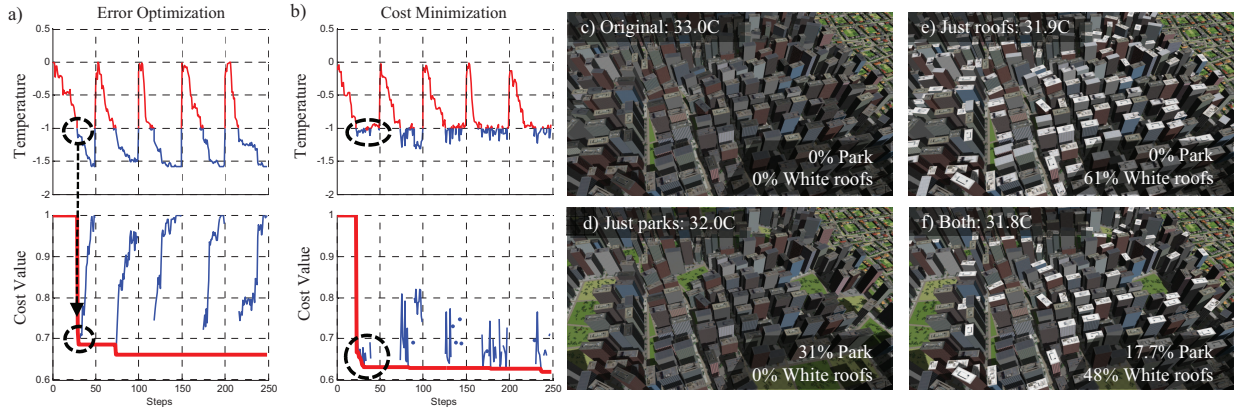

Fig. 12. **Inverse Temperature Design.** a-b) We show the behavior of the optimization for the solution e) of this figure: a) if our error optimization mode is used (i.e., optimize the temperature); b) if we use our cost minimization mode (i.e., temperature and cost optimization); c) the original model; d) altered model that achieves one degree reduction by introducing more parks; e) alternative model that achieves the same goal but uses white roofs to increase albedo; and f) a solution with both parks and white roofs (note the reduction in both).

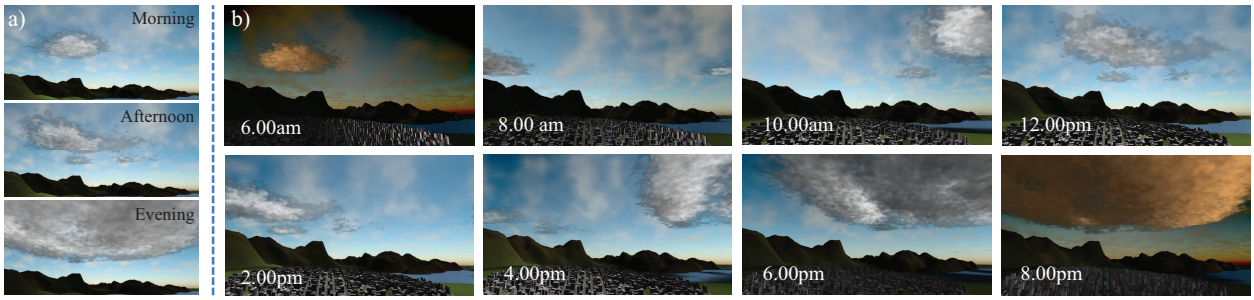

Fig. 13. **External Weather.** a) The user selects the desired cloud states. b) Our system is able to simulate the user-selected behavior.

As allowable changes to the initial model, we fix ω_l and ω_w and explore three alternative options for ω_p: change some urban land use to parks, paint some roofs to be of high albedo (e.g., White Roof Project [WRP-URL ; Santamouris 2014] is currently promoting this option), or perform both of the aforementioned changes.

First, we compare the standard acceptance ratio where one variable is optimized (Figure 12a) with our modified formula where we optimize a variable while we keep the cost to the minimum (Figure 12b). We run 5 chains for 50 steps with the original model as initial conditions and different $\beta \in (10, 650)$ (each chain is displayed after the other to analyze the behavior). In Figure 12a, the temperature is progressively reduced by the optimization method until it crosses the desired goal just once per energy level (the first cross of $\eta = -1$ is highlighted). After it crosses, the temperature is reduced further, but the cost is likely to increase as well. Thus, the best solution is found at one of these crosses. In Figure 12b, we present our cost minimization method. The temperature is reduced by the optimization, as before, but when the desired level is achieved $(\eta = [E(\Omega_0) - 1]$ Celsius), the optimization minimizes the cost. Both optimization modes alternate trying to find the goal temperature with the minimum cost. Note that this behavior could be understood as a multi-objective minimization where the solution lies in the Pareto frontier [Caramia and Dell'Olmo 2008]. We reach a frontier (it might have more than one since the space is highly nonlinear) where $E(*) \approx \eta$ and then the optimization explores to find a better solution staying close to it.

Further, Figure 12c shows the original model and Figure 12d shows the model with 31% more parks and 1° C reduction in temperature. Figure 12e shows an alternative option with 61% more white roofs and 1.1° C lower temperature. Figure 12f uses both more parks and white roofs to achieve 1.2° C reduction. Overall, the design tool enables quick exploration of options to achieve the desired weather/climatic change.

Incoming Weather.

Figure 13 shows an example of our external weather design. The user, instead of defining the general behavior of weather during a day (Figure 9), decides to use our external weather design to select a few interesting cloud configurations (Figure 13a) from the precomputed states (in our case, any of the $128 \cdot 10 \cdot 72$ precomputed states), the order, and the time period of simulation. By changing the boundary conditions, our system is able to simulate the user-defined behavior (Figure 13b).

7.3 Comparisons

To validate our simulator, we compare it to state-of-the-art weather modeling results and systems. We validate each of the three main components of our framework separately. To validate the first component, fundamental equations, we perform a comparison to well-known benchmark cases of a bubble of cold air and a bubble of warm air under prescribed conditions [Straka et al. 1993]. Figures 14 and 15 have 2D visualizations of such bubbles. In Fig-

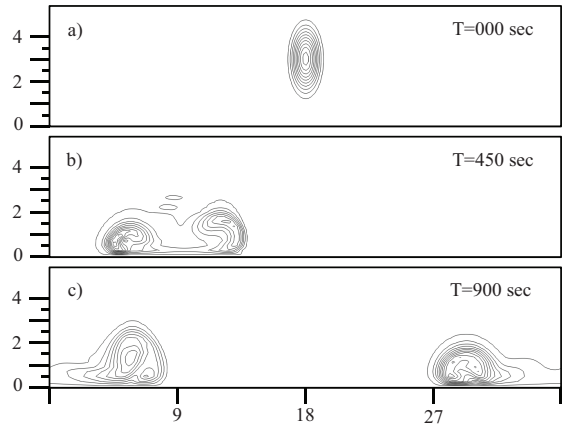

Fig. 14. **Cold Bubble.** Our evolution of potential temperatures similar to that of Wicker and Skamarock [2002]. Vertical axis is height and horizontal axis is spatial x-axis location (both in km). Simulation isolines at a) 0 seconds, b) 450 seconds, and c) 900 seconds.

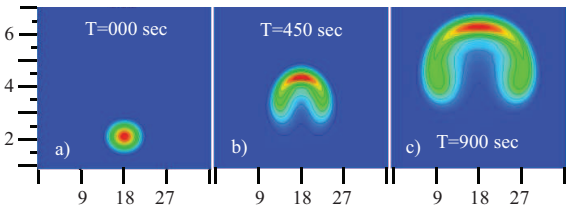

Fig. 15. **Warm Bubble.** Our simulation produces potential temperatures very similar to Ahmad and Lindeman [2007]. Note the axes and temporal sequence are the same as in Figure 8 of that paper.

ure 14a, an elliptical cold bubble ($-12.5C$) is placed in the center of the domain at a height of $3km$ using a grid cell size of $0.125km$ and a time step of $0.25s$. As should be the case, after $450s$ of simulation the bubble sinks and Kelvin-Helmholtz eddies are produced (Figure 14b). We impose wind of $20m/s$ which as per Wicker and Skamarock [2002] makes the test more stringent. At $900s$, as expected, the eddies have completed a half revolution around the domain and approach each other (Figure 14c). This behavior is very similar to Figure 2 of Wicker and Skamarock [2002], a classical reference. Figure 15 has a spherical warm bubble of $28C$ and at $2km$ above the ground, and same grid cell size and time step as in Figure 14. After $900s$, the bubble rises as should be the case as per Ahmad and Lindeman [2007].

To verify the second component, we compare our clouds and precipitation model to the WRF-ARW (Weather Research and Forecast Model – Advanced Research Version 3.6.1) and an expected cloud/rain formation process. Figures 16a-b show the computed value of water vapor q_v at different heights in a rural area and in an urban area using WRF and using our model – both models behave similarly. However, in a mesoscale model, WRF does not model clouds explicitly nor render them. Thus, we look to the literature on cloud dynamics. Cumulus clouds are formed by convergent-divergent zones producing powerful updrafts that form large vertical clouds and typically significant rainfall. We use our simulator to generate such a cumulus cloud adding a $3C$ warm bubble in the city center and successfully show its three main stages (Figure 16c-e): towering cumulus stage, mature stage, and dissipating stage. As

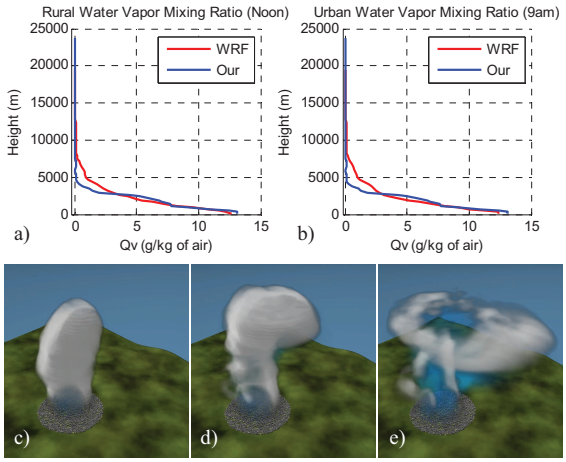

Fig. 16. **Cloud and Precipitation Comparison.** a-b) We show the water vapor mixing ratios for an urban and rural area using WRF's and our implementation. c-e) A temporal evolution of a cumulus cloud with our implementation.

expected, this large cloud forms a thunderstorm (e.g., see the rainfall depicted in Figure 16e) during the third stage that resembles a classic storm around a city [Bluestein and Parks 1983; Niyogi et al. 2011].

For the third component, we compare WRF's radiation and energy balance model with our model. The simulated domain consists of a simple circular city in the middle of an otherwise green terrain. As can be observed in Figure 17a, the relative temperature difference between urban and non-urban areas, defining the urban heat island, computed by the two models is almost identical. The shown curves are the difference between an East-West slice through the middle of the domain and an East-West slice through the southern part of the domain (not intersecting the city) as computed by each model. We also show the temperature evolution over time for a point in a rural area and in an urban area (Figure 17b-c), using both models. Our model again behaves very similarly to WRF.

It is important to highlight that weather forecasting is inherently nonlinear and by definition chaotic [Lorenz 1969]. Small changes to initial conditions (either prescribed or computed/interpolated by different model routines) can result in significant differences in the output. Even for two sophisticated models or for two runs with different physics options the model output can have significant differences. This is the basic motivation behind the so-called ensemble weather forecasting ([Barker et al. 2012]) used by the meteorological community to produce values such as a probability of precipitation. We highlight this to provide perspective on the similar (but not identical) output from the two models as a basis of verification and agreement.

Impact of Land Use.

Figure 18 shows the impact of land use on weather. Figure 18a shows the cloud coverage evolution over a 12 hour period for the same initial weather conditions while varying land use (Figure 18b). Note that when land use is almost homogenous, convection is not triggered and, despite the humidity, clouds do not form (see line 1 in Figure 18). Adding additional vegetation allows convection to produce clouds (line 2). Adding a city that presents a strong interface between land uses (line 3), foments cloud creation. Adding more complex structures (line 4) increases the chances to

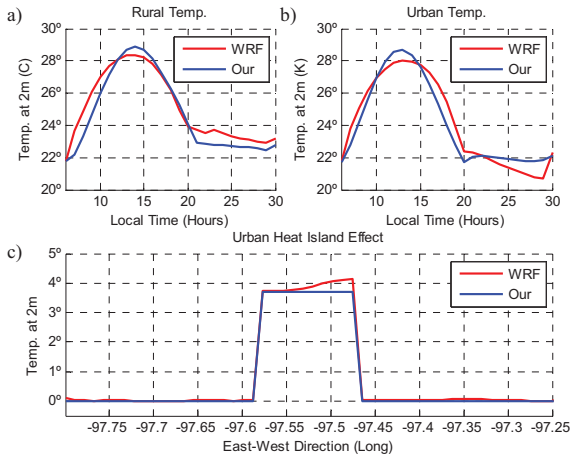

Fig. 17. **Radiation Model Comparison.** We compare WRF's radiation model to our radiation model: a-b) we show the temperature evolution over time for a point in a non-urban and in an urban portion; c) urban heat island effect as the temperature difference between an urban and non-urban 1D slice using each model. Our model behaves quite similar to WRF's.

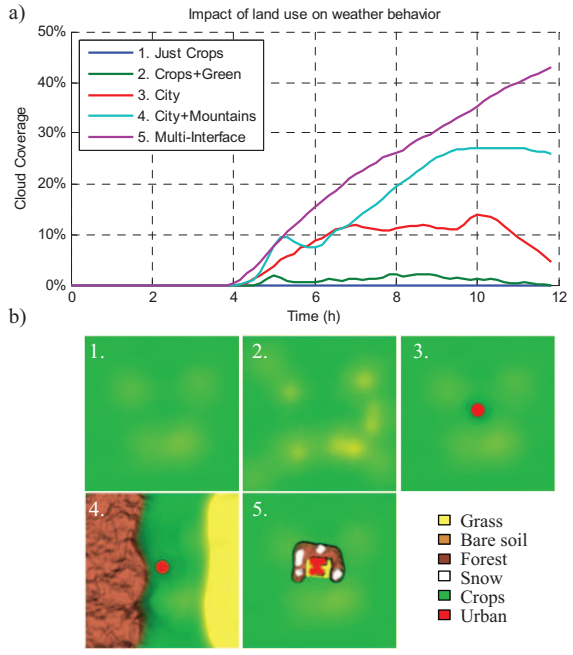

Fig. 18. **Impact of Land Use.** We show the impact of the land use on the weather behavior using the same initial weather conditions and modifying the land use. a) Cloud coverage over time for the five land uses shown in b).

create more compact and complex clouds. Note that the extent of the land use beyond a minimal threshold is not the main factor (line 5) instead, it is the different land use interfaces and associated mesoscale boundaries that initiates all weather phenomena.

Performance.

For a simulated volume of 50x50km and 25km high (divided into 50x50x56 grid cells), our 2D *XZ* system (Section 7.2) computes 81 minutes of weather in one second and our 3D system computes 1.7 minutes of weather in one second. In all cases, we use a simulation time step of $\Delta t = 1s$. Table I summarizes our performance on the aforementioned CPU and GPU. Our GPU implementation reaches over 101x with respect to real-time, and it is 68x faster than the CPU counterpart. As a reference, WRF is 7.1x faster than real time running on a compute server with 4 AMD Opteron 6176 12-core processors (450 GFLOP) – this maps to approximately 1.2x faster than real-time using our Intel Core i-7 (170 GFLOP). Note that this comparison is not straightforward since WRF is optimized to run on servers and is a globally compatible model with various formulations and options. For instance, a typical scenario for a WRF run would be using three nests of 27:9:3 resolution using a high-performance cluster (256-512 cores).

Table I. **Performance.** CPU uses 4 cores while GPU uses 2304 cores. See main text for additional details.

		Time per Time Step (ms)		Faster than Real-Time
		CPU	GPU	(GPU)
2DXY	Fund.	25.02	0.08	12195x
	+Clouds	62.98	0.16	6135x
	+Rad.	64.72	0.21	4878x
3D	Fund.	250.23	5.04	199x
	+Clouds	639.11	9.71	103x
	+Rad.	671.72	9.90	101x

8. CONCLUSIONS AND FUTURE WORK

We have presented a framework for realistic, physically-based weather simulation in urban procedural modeling. Our weather simulation is based on a non-hydrostatic weather model consisting of a set of nonlinear dynamical equations that govern atmospheric motions. It runs at super real-time rates (up to 4800 faster than real-time). We also present forward and inverse modeling weather design tools to control the simulation. Finally, we compare our system against the well-known state-of-the-art weather forecast results and systems. Altogether, our framework provides detailed and realistic weather phenomena without having to carefully script all meteorological phenomena.

For our current framework, we have identified several limitations and future work. First, since our focus is urban-scale our model cannot simulate weather phenomena that are formed on bigger or smaller scale; we will explore multi-resolution grids to address this and enhance the global design. Second, our microphysics model currently simulates warm-season cumulus clouds and rain. We will explore more complex models to add snow and hail. Third, we will include additional land use categories including modeling the effect of terrain height on the land use properties and weather grid variables. Fourth, we will explore the use of shared memory, dynamic parallelism, and streaming to enhance GPU performance. Fifth, we will explore physically based weathering of urban models, such as building and energy-uses, using our weather simulator.

ACKNOWLEDGMENTS

This research was partially funded by NSF CBET 1250232, NSF IIS 1302172, NSF AGS 1522494, NSF CAREER 0847472, NASA Earth System Science Fellowships, and a Google Research Award. We would also like to thank the Editor and the anonymous reviewers for their constructive suggestions.

REFERENCES

AHMAD, N. AND LINDEMAN, J. 2007. Euler solutions using flux-based wave decomposition. *International Journal for Numerical Methods in Fluids 54,* 1, 47–72.

ANDO, R., THUEREY, N., AND WOJTAN, C. 2015. A stream function solver for liquid simulations. *ACM Transactions on Graphics (TOG) 34,* 4, 53.

ARAKAWA, A. AND LAMB, V. R. 1977. Computational design of the basic dynamical processes of the ucla general circulation model. *Methods in computational physics 17,* 173–265.

ARYA, S. P. 1999. *Air pollution meteorology and dispersion*. Oxford University Press New York.

BARKER, D., HUANG, X.-Y., LIU, Z., AULIGNÉ, T., ZHANG, X., RUGG, S., AJJAJI, R., BOURGEOIS, A., BRAY, J., CHEN, Y., ET AL. 2012. The weather research and forecasting model's community variational/ensemble data assimilation system: Wrfda. *Bulletin of the American Meteorological Society 93,* 6, 831–843.

BLACKADAR, A. 1978. Modeling pollutant transfer during daytime convection. In *Symposium on Turbulence, Diffusion, and Air Pollution, 4 th, Reno, Nev*. 443–447.

BLINN, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. In *ACM SIGGRAPH Computer Graphics*. Vol. 16. ACM, 21–29.

BLUESTEIN, H. B. AND PARKS, C. R. 1983. A synoptic and photographic climatology of low-precipitation severe thunderstorms in the southern plains. *Monthly weather review 111,* 10, 2034–2046.

BOKELOH, M., WAND, M., AND SEIDEL, H.-P. 2010. A connection between partial symmetry and inverse procedural modeling. In *ACM Transactions on Graphics (TOG)*. Vol. 29. ACM, 104.

BOSCH, C., LAFFONT, P.-Y., RUSHMEIER, H., DORSEY, J., AND DRETTAKIS, G. 2011. Image-guided weathering: A new approach applied to flow phenomena. *ACM Transactions on Graphics 30,* 3.

BOUTHORS, A., NEYRET, F., MAX, N., BRUNETON, E., AND CRASSIN, C. 2008. Interactive multiple anisotropic scattering in clouds. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. ACM, 173–182.

CARAMIA, M. AND DELL'OLMO, P. 2008. *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms*. Springer Science & Business Media.

CHEN, F. AND DUDHIA, J. 2001. Coupling an advanced land surface-hydrology model with the penn state-ncar mm5 modeling system. part i: Model implementation and sensitivity. *Monthly Weather Review 129,* 4, 569–585.

CHEN, F., KUSAKA, H., BORNSTEIN, R., CHING, J., GRIMMOND, C., GROSSMAN-CLARKE, S., LORIDAN, T., MANNING, K. W., MARTILLI, A., MIAO, S., ET AL. 2011. The integrated wrf/urban modelling system: development, evaluation, and applications to urban environmental problems. *International Journal of Climatology 31,* 2, 273–288.

CHEN, Y., XIA, L., WONG, T.-T., TONG, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Visual simulation of weathering by γ-ton tracing. In *ACM Transactions on Graphics (TOG)*. Vol. 24. ACM, 1127–1133.

CITYENGINE. 2016. www.esri.com. Accessed: 2016-10.

COTTON, W. R., PIELKE SR, R., WALKO, R., LISTON, G., TREMBACK, C., JIANG, H., MCANELLY, R., HARRINGTON, J., NICHOLLS, M., CARRIO, G., ET AL. 2003. Rams 2001: Current status and future directions. *Meteorology and Atmospheric Physics 82,* 1-4, 5–29.

DOBASHI, Y., KANEDA, K., YAMASHITA, H., OKITA, T., AND NISHITA, T. 2000. A simple, efficient method for realistic animation of clouds. In *Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 19–28.

DOBASHI, Y., KUSUMOTO, K., NISHITA, T., AND YAMAMOTO, T. 2008. Feedback control of cumuliform cloud formation based on computational fluid dynamics. *ACM Transactions on Graphics (TOG) 27,* 3, 94.

DUNBAR, D. AND HUMPHREYS, G. 2006. A spatial data structure for fast poisson-disk sample generation. In *ACM Transactions on Graphics (TOG)*. Vol. 25. ACM, 503–508.

DURRAN, D. R. 1989. Improving the anelastic approximation. *Journal of the atmospheric sciences 46,* 11, 1453–1461.

DURRAN, D. R. 2013. *Numerical methods for wave equations in geophysical fluid dynamics*. Vol. 32. Springer Science & Business Media.

EBERT, D. S. 2003. *Texturing & modeling: a procedural approach*. Morgan Kaufmann.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 15–22.

GARCIA-DORADO, I., G ALIAGA, D., AND V UKKUSURI, S. 2014. Designing large-scale interactive traffic animations for urban modeling. In *Computer Graphics Forum*. Vol. 33. Wiley Online Library, 411–420.

GARG, K. AND NAYAR, S. K. 2006. Photorealistic rendering of rain streaks. *ACM Transactions on Graphics (TOG) 25,* 3, 996–1002.

GERO, A., PITMAN, A., NARISMA, G., JACOBSON, C., AND PIELKE, R. 2006. The impact of land cover change on storms in the sydney basin, australia. *Global and Planetary Change 54,* 1, 57–78.

GOTANDA, Y., KAWASE, M., AND KAKIMOTO, M. 2015. Real-time rendering of physically based optical effects in theory and practice. In *ACM SIGGRAPH 2015 Courses*. ACM, 23.

HARRIS, M. J., BAXTER, W. V., SCHEUERMANN, T., AND LASTRA, A. 2003. Simulation of cloud dynamics on graphics hardware. In *Proceedings of Graphics Hardware*. Eurographics Association, 92–101.

HARRIS, M. J. AND LASTRA, A. 2001. Real-time cloud rendering. In *Computer Graphics Forum*. Vol. 20. Wiley Online Library, 76–85.

HASTINGS, W. K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika 57,* 1, 97–109.

HOLTON, J. R. AND HAKIM, G. J. 2012. *An introduction to dynamic meteorology*. Vol. 88. Academic press.

JANTZ, C. A., GOETZ, S. J., DONATO, D., AND CLAGGETT, P. 2010. Designing and implementing a regional urban modeling system using the sleuth cellular urban model. *Computers, Environment and Urban Systems 34,* 1, 1–16.

KAJIYA, J. T. AND VON HERZEN, B. P. 1984. Ray tracing volume densities. In *ACM SIGGRAPH Computer Graphics*. Vol. 18. ACM, 165–174.

KESSLER, E. 1969. *On the distribution and continuity of water substance in atmospheric circulation*. American Meteorological Society.

KNISS, J., PREMOZE, S., HANSEN, C., SHIRLEY, P., AND MCPHERSON, A. 2003. A model for volume lighting and modeling. *Visualization and Computer Graphics, IEEE Transactions on 9,* 2, 150–162.

LORENZ, E. N. 1969. Atmospheric predictability as revealed by naturally occurring analogues. *Journal of the Atmospheric sciences 26,* 4, 636–646.

LUCKE, T. AND NICHOLS, P. W. 2015. The pollution removal and stormwater reduction performance of street-side bioretention basins after ten years in operation. *Science of The Total Environment 536,* 784 – 792.

MACKLIN, M. AND MÜLLER, M. 2013. Position based fluids. *ACM Transactions on Graphics (TOG) 32,* 4, 104.

MANOCHA, D. AND LIN, M. C. 2012. Interactive large-scale crowd simulation. In *Digital Urban Modeling and Simulation*. Springer, 221–235.

MARSHALL, J. S. AND PALMER, W. M. K. 1948. The distribution of raindrops with size. *Journal of meteorology 5,* 4, 165–166.

MESINGER, F., DIMEGO, G., KALNAY, E., MITCHELL, K., SHAFRAN, P. C., EBISUZAKI, W., JOVIC, D., WOOLLEN, J., ROGERS, E., BERBERY, E. H., ET AL. 2006. North american regional reanalysis. *Bulletin of the American Meteorological Society 87*, 3, 343–360.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics 21*, 6, 1087–1092.

MICHALAKES, J. AND VACHHARAJANI, M. 2008. Gpu acceleration of numerical weather prediction. *Parallel Processing Letters 18*, 04, 531–548.

MIELIKAINEN, J., HUANG, B., WANG, J., HUANG, H.-L. A., AND GOLDBERG, M. D. 2013. Compute unified device architecture (cuda)-based parallelization of wrf kessler cloud microphysics scheme. *Computers & Geosciences 52*, 292–299.

MIYAZAKI, R., YOSHIDA, S., DOBASHI, Y., AND NISHITA, T. 2001. A method for modeling clouds based on atmospheric fluid dynamics. In *Computer Graphics and Applications*. IEEE, 363–372.

MONIN, A. AND OBUKHOV, A. 1954. Basic laws of turbulent mixing in the surface layer of the atmosphere. *Contrib. Geophys. Inst. Acad. Sci. USSR 151*, 163–187.

MULLANEY, J., LUCKE, T., AND TRUEMAN, S. J. 2015. A review of benefits and challenges in growing street trees in paved urban environments. *Landscape and Urban Planning 134*, 157–166.

MUSIALSKI, P., WONKA, P., ALIAGA, D. G., WIMMER, M., GOOL, L., AND PURGATHOFER, W. 2013. A survey of urban reconstruction. In *Computer graphics forum*. Vol. 32. Wiley Online Library, 146–177.

NEBEKER, F. 1995. *Calculating the weather: Meteorology in the 20th century*. Vol. 60. Academic Press.

NISHITA, T., DOBASHI, Y., AND NAKAMAE, E. 1996. Display of clouds taking into account multiple anisotropic scattering and sky light. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 379–386.

NISHITA, T., IWASAKI, H., DOBASHI, Y., AND NAKAMAE, E. 1997. A modeling and rendering method for snow by using metaballs. In *Computer Graphics Forum*. Vol. 16. Wiley Online Library, C357–C364.

NIYOGI, D., PYLE, P., LEI, M., ARYA, S. P., KISHTAWAL, C. M., SHEPHERD, M., CHEN, F., AND WOLFE, B. 2011. Urban modification of thunderstorms: an observational storm climatology and model case study for the indianapolis urban region*. *Journal of Applied Meteorology and Climatology 50*, 5, 1129–1144.

NOILHAN, J. AND PLANTON, S. 1989. A simple parameterization of land surface processes for meteorological models. *Monthly Weather Review 117*, 3, 536–549.

OKE, T. R. 2002. *Boundary layer climates*. Routledge.

OVERBY, D., MELEK, Z., AND KEYSER, J. 2002. Interactive physically-based cloud simulation. In *Computer Graphics and Applications*. IEEE, 469–470.

PARISH, Y. I. AND MÜLLER, P. 2001. Procedural modeling of cities. In *Computer graphics and interactive techniques*. ACM, 301–308.

PIELKE, R., STOKOWSKI, D., WANG, J.-W., VUKICEVIC, T., LEONCINI, G., MATSUI, T., CASTRO, C. L., NIYOGI, D., KISHTAWAL, C. M., BIAZAR, A., ET AL. 2007. Satellite-based model parameterization of diabatic heating. *Eos, Transactions American Geophysical Union 88*, 8, 96–97.

PIELKE SR, R. A. 2013. *Mesoscale meteorological modeling*. Vol. 98. Academic press.

PURSER, R. AND LESLIE, L. 1988. A semi-implicit, semi-lagrangian finite-difference scheme using hligh-order spatial differencing on a non-staggered grid. *Monthly Weather Review 116*, 10, 2069–2080.

RANDALL, D. A. AND HUFFMAN, G. J. 1980. A stochastic model of cumulus clumping. *Atmospheric Sciences 37*, 9, 2068–2078.

ROLAND, S. 2000. Meteorology for scientists and engineers. *Brooks/Cole*.

SANTAMOURIS, M. 2014. Cooling the cities–a review of reflective and green roof mitigation technologies to fight heat island and improve comfort in urban environments. *Solar Energy 103*, 682–703.

SCHMID, P. E. AND NIYOGI, D. 2013. Impact of city size on precipitation-modifying potential. *Geophysical Research Letters 40*, 19, 5263–5267.

SEWALL, J., WILKIE, D., AND LIN, M. C. 2011. Interactive hybrid simulation of large-scale traffic. In *ACM Transactions on Graphics (TOG)*. Vol. 30. ACM, 135.

SKAMAROCK, W. C., KLEMP, J. B., DUDHIA, J., GILL, D. O., BARKER, D. M., WANG, W., AND POWERS, J. G. 2005. A description of the advanced research wrf version 2. Tech. rep., DTIC Document.

SOLECKI, W. D., ROSENZWEIG, C., PARSHALL, L., POPE, G., CLARK, M., COX, J., AND WIENCKE, M. 2005. Mitigation of the heat island effect in urban new jersey. *Global Environmental Change Part B: Environmental Hazards 6*, 1, 39–49.

SOONG, S.-T. AND OGURA, Y. 1973. A comparison between axisymmetric and slab-symmetric cumulus cloud models. *Journal of the Atmospheric Sciences 30*, 5, 879–893.

STEINHOFF, J. AND UNDERHILL, D. 1994. Modification of the euler equations for vorticity confinement: application to the computation of interacting vortex rings. *Physics of Fluids 6*, 8, 2738–2744.

STRAKA, J., WILHELMSON, R. B., WICKER, L. J., ANDERSON, J. R., AND DROEGEMEIER, K. K. 1993. Numerical solutions of a non-linear density current: A benchmark solution and comparisons. *International Journal for Numerical Methods in Fluids 17*, 1, 1–22.

STULL, R. B. 1988. *An introduction to boundary layer meteorology*. Vol. 13. Springer Science & Business Media.

TALTON, J. O., LOU, Y., LESSER, S., DUKE, J., MĚCH, R., AND KOLTUN, V. 2011. Metropolis procedural modeling. *ACM Transactions on Graphics (TOG) 30*, 2, 11.

TETENS, O. 1930. Uber einige meteorologische begriffe. *Z. Geophys. 6*, 297–309.

US-CENSUS. 2016. www.census.gov. Accessed: 2016-10.

VANEGAS, C. A., ALIAGA, D. G., WONKA, P., MÜLLER, P., WADDELL, P., AND WATSON, B. 2010. Modelling the appearance and behaviour of urban spaces. In *Computer Graphics Forum*. Vol. 29. Wiley Online Library, 25–42.

VANEGAS, C. A., GARCIA-DORADO, I., ALIAGA, D. G., BENES, B., AND WADDELL, P. 2012. Inverse design of urban procedural models. *ACM Transactions on Graphics (TOG) 31*, 6, 168.

WICKER, L. J. AND SKAMAROCK, W. C. 2002. Time-splitting methods for elastic models using forward time schemes. *Monthly Weather Review 130*, 8, 2088–2097.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. *Instant architecture*. Vol. 22. ACM.

WRENNINGE1, M. AND BIN ZAFAR, N. 2011. Production volume rendering. In *ACM SIGGRAPH 2011 Courses*. ACM, 71.

WRP-URL. www.whiteroofproject.org. The White Roof Project. Accessed: 2016-10.

YUAN, C., LIANG, X., HAO, S., QI, Y., AND ZHAO, Q. 2014. Modelling cumulus cloud shape from a single image. In *Computer Graphics Forum*. Vol. 33. Wiley Online Library, 288–297.

ZHANG, C. L., CHEN, F., MIAO, S. G., LI, Q. C., XIA, X. A., AND XUAN, C. Y. 2009. Impacts of urban expansion and future green planting on summer precipitation in the beijing metropolitan area. *Journal of Geophysical Research: Atmospheres (1984–2012) 114*, D2.

Received Xxxxx XXXX; accepted Xxxx XXXX