Weather simulation

TERIIHOANIA Joan Heimanu
joan.teriihoania@etu.umontpellier.fr

27 November 2021

Abstract

This paper presents the methodologies used to simulate the weather over a given world heightmap. The solution presented is based on fluid simulation techniques and real-world weather patterns.

Introduction

Weather simulation is an every-day challenge for meteorologists as well as computer scientists all around the globe. Being able to predict future weather events can help save lives as well as understand the current stakes of climate change. Today, most weather models developed over the years have achieved a level of sophistication and precision never seen before and can predict with extreme accuracy the temperature, rain, and wind of areas a week in advance. However, they are very complex and resource intensive. This paper will introduce a simpler and less resource hungry model to weather simulation.

Mathematical definitions

1 - Matrices

In this paper, matrices will be expressed as 2d-grids of $n = c \times l$ cells, c columns and l lines and can vary in two types.

1.1 - Static

These matrices are as any conventional mathematical matrix and are noted as M^s.

1.2 - Relative

A relative matrix M^r is a matrix with:

$$c = l$$

$$\left\lfloor \frac{c}{2} \right\rfloor = \frac{c}{2}$$

The elements coordinates (x, y) are relative over the center $(0,0)$ of the base matrix M^s where:

$$M^r_{x-\frac{c-1}{2},y-\frac{l-1}{2}} = M^s_{x,y}$$

Principles

1 – World projection

To be able to easily represent a map into a usable format for us to compute, the *Mercator projection* can be used.

2 – (Point/cell)-based simulation

This paper will detail the implementation of a weather simulation using a point-based system over a 2d-field where each point contains its own atmospheric properties that will interact with its neighbors.

3 – Atmospheric properties

3.1 – Wind

It is a directed force that moves a certain quantity of a property in the environment from one place to another. This relation is double-sided, meaning that wind both influences and is influenced by its environment. However, in addition to that, wind is also a movable property, meaning that it too can be moved by itself.

3.2 – (Atmospheric) Humidity

It is the amount of water droplets present in a certain region of the atmosphere and is usually represented as a percentage. It mostly influences clouds and rain.

3.3 – Surface humidity

It is the amount of water saturation on the ground and is represented as a percentage, like the atmospheric humidity. It mostly influences the growth of vegetation and the amount of evaporation. Note that this property should always be maxed out when a point is considered an infinite body of water such as the sea.

3.4 – Temperature

It is represented in Celsius degrees and should be capped to extremums of arbitrary values.

3.5 – Vegetation

It is the amount of vegetation and is represented as a percentage. It requires surface humidity to grow and will decrease when none is present.

Implementation

1 – Classes

1.1 - Map

A static matrix noted Map representing a world with each point (x, y) containing a $Cell$.

1.2 - Cell

Noted $Cell$, it contains all the various information of a point in the world it's in including Surface humidity $s \in [0,1]$, humidity $h \in [0,1]$, temperature t, height $h \in [0,1]$, vegetation $v \in [0,1]$ and wind as a vector (x, y) representing the direction and force of the wind.

2 – Properties movement over the grid

Our model revolves around a 2d-grid of points, which uses integer coordinates, but the wind vector uses float-based coordinates. So, to calculate the movement of the various movable properties (humidity, temperature, wind) of a Cell g, we must first calculate the arrival point based on the wind vector and then sample the surrounding grid points which will receive a part of the moved property depending on its proximity to the arrival point. Therefore, first, we calculate the arrival coordinates e of our point g with its wind properties.

$$e = (x, y) = \left(g_x + g_{wind_x}, g_x + g_{wind_y}\right)$$

Then, we take the four closest points cp of the arrival coordinates e.

$$cp = \begin{bmatrix} b & c \\ a & d \end{bmatrix} = \begin{bmatrix} (\lfloor e_x \rfloor, \lceil e_y \rceil) & (\lceil e_x \rceil, \lceil e_y \rceil) \\ (\lfloor e_x \rfloor, \lfloor e_y \rfloor) & (\lceil e_x \rceil, \lfloor e_y \rfloor) \end{bmatrix}$$

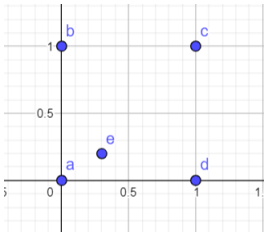

Figure 1 - Graphical representation of closest cardinal points

We can then calculate each point's proximity coefficient p (used to weight the distribution of the properties) to e with:

$$diff(l) = (x, y) = \left(|l_x - e_x|, |l_y - e_y|\right)$$

$$\forall c \in cp, p_c = \frac{1 + \left(1 - \left(diff(c)_x + diff(c)_y\right)\right)}{4}$$

3 – Wind over environment

We also have to calculate the wind direction and force based on the environment (temperature, height and humidity). For that, we will need a relative matrix N^r of the neighbors of our point a with the values we want (temperature, height or humidity) with $N^r_{x,y} = Map_{a_x - x, a_y - y}$. With it, we calculate the matrix of differences M^{diff} with $M^{diff}_{x,y} = N^r_{x,y} - N^r_{0,0}$. Then, we calculate the directed vector:

$$\begin{pmatrix} \frac{\left(\frac{M^{diff}_{-1;-1}}{2} + M^{diff}_{-1;0} + \frac{M^{diff}_{-1;1}}{2}\right)}{2} - \frac{\left(\frac{M^{diff}_{1;-1}}{2} + M^{diff}_{1;0} + \frac{M^{diff}_{1;1}}{2}\right)}{2} \\ \frac{\left(\frac{M^{diff}_{-1;-1}}{2} + M^{diff}_{0;-1} + \frac{M^{diff}_{1;-1}}{2}\right)}{2} - \frac{\left(\frac{M^{diff}_{-1;1}}{2} + M^{diff}_{0;1} + \frac{M^{diff}_{1;1}}{2}\right)}{2} \end{pmatrix}$$

Note that this vector represents only the desired wind direction and force of the calculated environment property (temperature, ...). To get the effective wind vector of our point, we have to mix all vectors over the different properties weighted over arbitrary coefficients based on their importance. We recommend setting the coefficients as $\{temperature: 0.1, height: 0.2, humidity: 0.1\}$

4 – Water cycle

4.1 - Evaporation

This phase consists of the conversion of surface humidity into (atmospheric) humidity. This occurs when the temperature at a certain point a exceeds a certain threshold t_{ev} with the amount of evaporated humidity ev:

$$a_{surface\ humidity} \times \left(\frac{a_{temperature} - 20}{100} \times (1 + a_{vegetation} \times ev_{coef})\right)$$

With $ev_{coef} \in [0,1]$, an arbitrary coefficient to control the vegetation's evaporation's speed's increase.

4.2 – Cloud

For simplicity's sake, the mechanisms behind the creation of clouds have been crudely simplified. In our model, this phenomenon occurs in two phases.

4.2.1 – Atmospheric cooling

When a humidity threshold h_{ac} is reached, the temperature starts decreasing depending on the current atmospheric humidity at a point a and an arbitrary coefficient ac_{coef} with the amount of temperature decrease:

$$ac = a_{humidity} \times ac_{coef}$$

4.2.2 – Condensation

Once the temperature drops below a certain threshold t_{con} and humidity is still above another one h_{con}, condensation occurs, and clouds start to appear.

4.3 - Rain

When the temperature drops below t_{rain} with $t_{rain} < t_{con}$ and humidity is higher than h_{rain} with $h_{rain} > h_{con}$, a certain amount $rain$ is converted from (atmospheric) humidity to surface humidity based on an arbitrary coefficient $rain_{coef}$ and the temperature and humidity at the point a.

$$rain = a_{humidity} \times \left(1 - \frac{\max(a_{temperature} + 10, 40)}{60}\right) \times rain_{coef}$$

5 – Vegetation growth

This growth is a process where surface humidity is converted into vegetation. For simplicity's sake, the converted humidity will be completely lost to vegetation once it is converted. This happens very slowly, as such the growing $veg_{growcoef}$ and decreasing $veg_{deccoef}$ vegetation coefficient must be low (≤ 0.01). Considering a point a, when $a_{surface\ humidity} > 0$, growth occurs, and the amount converted will be:

$$veg_{grow} = a_{surface\ humidity} \times veg_{growcoef}$$

When $a_{surface\ humidity} = 0$, decrease occurs, and plants will start dying, losing the converted humidity. The amount lost will be:

$$veg_{dec} = a_{vegetation} \times veg_{deccoef}$$

6 – Natural heating and cooling

We discussed every effect of temperature over the environment but haven't talked about how temperature fluctuates in the first place. Natural heating occurs with the presence of the sun during the day and natural cooling in its absence.

6.1 – Sun position

The position of the sun noted sun_x and sun_y over the map depends on the date and time. At 00:00:00, $sun_x = 0$ and at 23:59:59, $sun_x = c - 1$. This implements the day-night cycle, but to implement the seasonal-cycle, we have to act on the sun_y position as well. For that, we have to supply a list of the sun *absolute* positions depending on the month (through the use of a table or a function). When we talk about absolute sun position, we talk of the sun's position over the year from the North to the South extremities of its cycle. It can go over the poles or even not go over them at all.

As an example, here is a set of absolute positions we found that models the Earth.

Month	$asun_y^{month}$
January	1
February	0.84
March	0.68
April	0.5
May	0.34
June	0.18
July	0
August	0.16
September	0.34
October	0.5
November	0.66
December	0.82

Now that we have the absolute values, we can weight them depending on the day and calculate sun_y as:

$$(l - pole \times 2) \times \left(asun_y^{m-1} + \left(\frac{m}{d} \times now\right)\right) + pole$$

With m, the number of the current month, $pole$ the size of the poles where $l - 1 \geq pole \geq 0$, $diff = asun_y^m - asun_y^{m-1}$, d the number of days in the current month and now the number of days since the beginning of the month.

Problems

1 – World map projection inaccuracy

While using the *Mercator projection* facilitates the representation of a map into a 2d-grid for easy computation, this comes with disadvantages. As

we try to represent a sphere into a rectangular plane, information is warped.

1.1 – Size difference

This has the consequence of stretching or compacting certain regions, consequence that, for the sake of simplicity, isn't considered in the simulation. This renders the projection of the results of our simulation into a sphere quite complex.

1.2 – Relative coordinates

Another consequence is that, unlike a sphere, rectangles have an end. This needs to be considered when we calculate the point of arrival arr.

As you can see, if we were to implement this approach as is, we would run into a problem where the indices/coordinates (x, y) could potentially go outside of our matrix. As such, adding support for such out-of-bound coordinates must be done and a simple solution to that is the following.

If we have a point p of coordinates (x, y) in a matrix M of c columns and l lines, there are two situations we can find ourselves into.

1.2.1 – Horizontally out-of-bound

The coordinates given exceeds the limits of our matrix in the x-axis. If $x < 0$, then $x' = c - |x|$ and if $x \geq c$ then $x' = x - c$.

1.2.2 – Vertically out-of-bound

Same as above but for the y-axis. If that occurs, we will have $x' = \frac{c}{2} + x$. If $y < 0$, then $y' = 0$ and if $y \geq l$ then $y' = l - 1$.

2 – Performance

As of now, this model is very simple, a lot of principles have been kept out of the computation for future iterations or to keep the complexity of each simulation cycle (which is currently $O(n)$) as low as possible.

Improvements

1 – 3d-grid representation

One of the improvements that could be implemented into a new iteration could be a 3d-grid representation of the world. Although resource intensive, it would increase realism and enable the occurrence of more complex weather patterns such as vortexes or cyclones.

2 – Snow

As of now, this model does not support cold-type environment. This is a huge loss in realism as seasonal temperature changes can affect water supplies, rivers, and such water bodies. This would enable the occurrence of complex weather behavior like the drought or the rain season.

3 – Dynamic water mechanism

For now, there are two simplifications made into the water mechanisms of the simulation.

3.1 – Infinite Sea

The sea has an infinite amount of water, this is a simplification made to kickstart the simulation as there is no body of water at the first cycle.

3.2 – Water movement

There is no way for surface humidity aka water, other than evaporation, to move which is hugely unrealistic. This can be improved by implement water physics through the reuse of the wind movement mechanics. This would enable the creation of other types of water bodies such as lakes and rivers and supply continental regions with.

5 – Vegetation effects

As of now, once vegetation has grown, it has no actual effects over the environment. This can be improved by implementing plants influence.

5.1 – Water consumption

One effect is the plant's water consumption to maintain themselves. This effect could even entirely replace the decrease in vegetation details higher in this paper.

5.2 – Increase humidity capacity

In this iteration, humidity converted is lost and cannot be retrieved which is something completely unrealistic. This can be improved by integrating the vegetation as a new source of humidity, one which is more volatile than normal bodies of water.

5.3 – Shades (temperature cooling)

However, if we were to implement vegetation as a more volatile source of humidity for evaporation, this would only result into speeding up the evaporation process, emptying all surface humidity, resulting in destroying vegetation and creating an erratic cycle. One solution to this would be to decrease the amount of evaporation

over the surface humidity proportionally to the amount of vegetation. This will make plants able to consume rained water while losing some to vegetation's water evaporation, creating a positive intake for rain and even enabling the creation of swamps and water-type biomes.

Applications

You can find the results of this model implemented in a NodeJS application over 6 533 cycles outputted as two speedup videos hosted on YouTube.

Satelite map and Temperature map

Appendices

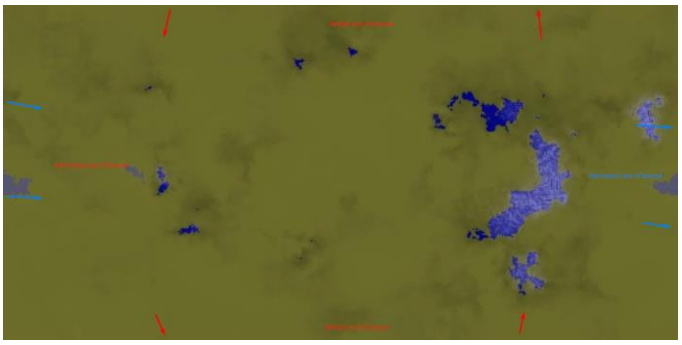

Figure 2 - Out of bound coordinates visualization