A fast, simple method to render sky color using gradients maps

Jesús Alonso Abad

Student at the Faculty of Computer Science, University of Burgos
e-mail: jaa0023@alu.ubu.es

October 9, 2006

Abstract

As new rendering technologies appear, complex yet accurate methods for visual simulation are used. This can mean a performance overkill though, as such accuracy isn't required often.

It's important to keep an eye in old hardware and other not-so-capable devices (PDAs, cell phones...), as well as performance-critical applications, which may prefer better performance by means of lowering the visual accuracy.

Here a method to obtain realistic results in such cases is presented through a simple but efficient method to render the sky dome gradients.

1 INTRODUCTION

Photorealism has become a demanded feature lately in many fields like games, simulation, or just marketing demonstrations. There has been a lot of research in nature effects simulation, such as clouds, rain, fog... and so a lot of methods appeared. Regarding to sky color rendering, very accurate models have come out from these researches, but usually accompanied by a performance loss.

As we can't always focus on high-end hardware, it's necessary to find some fast methods to get accurate results.

With this spirit, this method is presented as an accurate way to render skies, while using a very low amount of memory and CPU usage with the help of bitmaps.

2 METHOD DESCRIPTION

The basic idea is to use a sky dome and map the sky colors onto it. Let's explain each of these elements in detail.

2.1 The sky gradients map

As a starting point, a function like this should be defined:

$$f(a,t)=(r,g,b) \quad (1)$$

Where *a* means *altitude* (ranging from 0º to 90º –horizon to zenith), and *t* is the *day time* in a range of [0, 1]. *(r, g, b)* are the resulting color components, returned by that function.

This function calculates the base sky color depending on the altitude and day time, obtaining a sequence of sky gradients. If we sample the variables, we obtain a discrete number of colors, which can be mapped to a bitmap image. Many models used to simulate the sky color can generate these values. Of course, these values can be sampled from real-life through pictures, or even adjusted manually if some artistic touch is desired.

This map will be used, as described later, to map the gradients on the sky dome according to the day time.

2.2 The sky dome

The only requirements of the sky dome is to have a uniform U texture coordinate and that its V texture coordinate map exactly to its altitude value respect the dome center, as shown in the figure below:

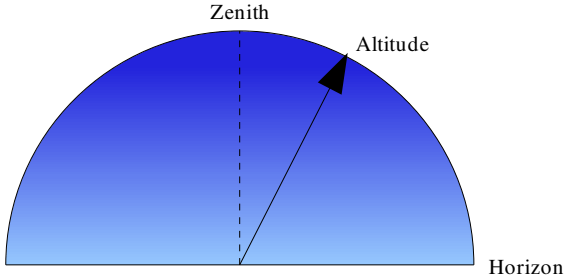

Illustration 1: Sky dome mapping

The idea behind this is that simply by modifying its U coordinate, the whole sky gradient will reflect the changes accordingly.

3 IMPLEMENTATION

Once presented the fundamentals of this method, some implementation notes will be presented.

3.1 Minimum suggested requirements

In order to obtain acceptable results, these elements are considered a requirement.

First of all, the sky dome should be a geodesic hemisphere with a medium tessellation. A full sphere will be better because of the terrain surface limitations in computer graphics, though if an infinite surface could be used, it's suggested to just use a hemisphere.

Second, the map should have a decent resolution. It's been proved that a 16x16 map gives nice results, though 32x32 or even 64x64 are suggested to have a good control over the gradient and the time steps.

Third and last, linear filtering for the texture is highly suggested, as it provides smooth gradients as well as smooth transitions between them. Texture wrapping is also suggested to achieve nice cyclic transitions between days (this is, horizontally), while texture clamping is suggested to avoid artifacts in the zenith and horizon points (vertically).

3.2 The sky dome

Any construction method used to build the sky dome is enough, with the only requirement that, as mentioned above, the texture coordinates should have the form (0, V). V is calculated as the altitude angle, and 0 is the suggested value, so the starting day-time matches the 0h00m gradient in the map.

3.3 Performance improvements

First attempts for this method were done by iteratively modifying the texture coordinates of the dome. This meant having to rebuild the whole geometry every time the day-time changed. However, some improvements can be done in this aspect.

The first of them is to use the texture matrix operations (specifically the translation operation) to go through time instead of recalculating the coordinates. This meant a high speed improvement that shouldn't be overlooked, as the hardware now does a single operation that we did dozens of times every frame through software.

As now the dome is not needed to be changed anymore, it can be stored in a vertex array or a hardware buffer so that it can be dumped in a single operation into the GPU (if present).

3.4 Final quality improvements

Light absorption

Though the resulting dome can feature an incredible quality, it may look too uniform , specially at dusk and dawn.

The atmosphere absorbs light power incrementally, the same way fog or water does. It's hard to notice when the sun is high, but at a low altitude, the amount of air the light must go through is noticeable high, so the sky color is darker at the opposite side to the sun.

This is represented in Illustration 2, where the light crossing the *A* segment suffers much more absorption than *B*. This effect is depicted in Illustration 3.

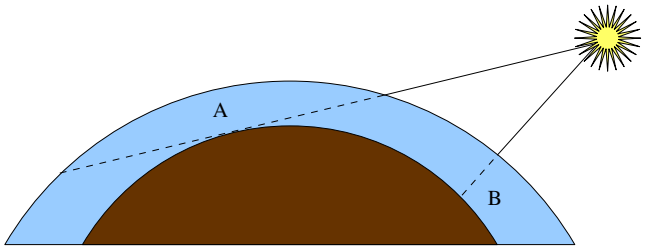

Illustration 2: Different light trajectories

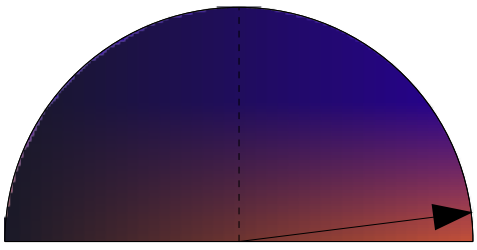

Illustration 3: Atmosphere darkening at the sunset

There are many possible ways to perform this operation. One of them using a shader to desaturate

and darken the sky color. It is possible as well to achieve a similar effect by using multitexturing, texture matrix operations and an additional UV map in the dome, though the color won't be desaturated. Which method to use depends on the balance between performance and minimum requirements.

Sun glow

The sun glow produced by sunlight inscatter in the sky dome can be easily done following an approach similar to the taken for the light absorption phenomena mentioned above.

Atmosphere composition

The sky color also depends on the air composition, humidity, turbidity, floating particles... All of these modify the final sky color.

The same as above, there are two ways to simulate this: Through shaders, that offer a finer control over the final result, and multitexturing, that is supported by a wider range of hardware.

Sky translucency

The sky becomes opaque (in relative terms) because of the light scattering. When an intense light goes through it, other bright objects are harder to see, such as the stars during mid-day.

This can be easily simulated by adding an alpha channel to the sky gradients map, so that the dark areas of the sky at night can be translucent or even transparent.

4 EXAMPLES

Illustration 5 shows a full day sequence, where the changing sky colors can be easily seen. The sequence ranges from night to next dusk, passing through dawn, sunrise, morning and sunset. The sky colors map was built from different real pictures taken from clear skies at Spain.

Illustration 6 shows the effect of the shader performing the task to simulate light absorption. The shader woks on the sky dome color and translucency, taking the incoming sky color and the light position to calculate the resulting color. When the sun is at a low height, the sky at the opposite side will be desaturated and become more translucent.

In the Illustration 7 the effects of the sun glow shader can be seen. This shader works in a similar manner to the light absorption shader, but in this case it simply enhances the light power around the sun in the sky color. Additional postfiltering operations would complete this operation simulating the strong light effects on the camera lenses.

Illustration 8 pictures the sky translucency functionality achieved with this method. The star field can be seen through the dark sky at the starting night. Note the more dense atmosphere (and thus, more opaque) near the horizon, while it keeps almost transparent near the zenith. This effect is stronger as the light absorption (see above) is stronger.

An additional subsystem was developed to manage the base haze. Illustration 9 depicts the effects of this subsystem. It's also image-based, and also support density through alpha channels. As can be seen in the pictures, its density increases along the day (because of water evaporation thanks to the sun light). This subsystem is not definite though, as the color and density also varies with other factors like overcast skies, pollution, etcetera, not discussed here.

5 CONCLUSION

As has been stated here, most of the complex models to render sky gradients can be simplified some way, for the sake of performance boost and a wider variety of hardware capable of running them.

Early performance tests show a loss of about 65% frames per second after enabling the whole system, as can be seen in the following chart. These tests were performed on an undebugged prototype, which may cause most of the performance loss, as well as halving the filling of the screen when the sky is not rendered, which is probably a determining factor.

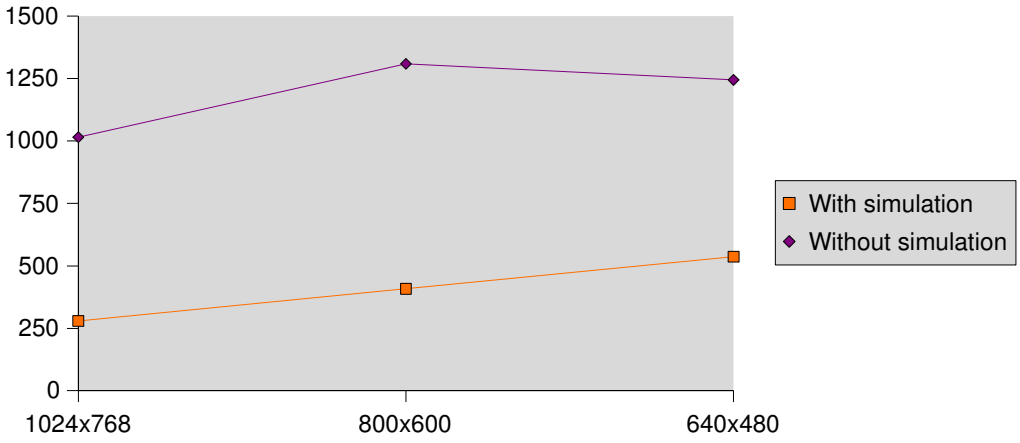

Illustration 4: Performance tests

Though the results of this method can be stunning, there are still possible improvements and research ways, such as the problem of adding support for multiple light sources (moon).

Acknowledgment

I would like to thank the Virtual Terrain Project crew (www.vterrain.org) for collecting a large amount of links to resources regarding this subject, as well as everyone who published papers about their research in this field, for the great inspiration and ideas.

References

[1] Ralf Stokholm Nielsen. *Real time rendering of atmospheric scattering effects for flight* simulators. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, September 2003.

[2] A. J. Preetham, Peter Shirley and Brian Smits. *A practical analytic model for daylight.* In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 91-100. ACM Press/Addison-Wesley Publishing Co., 1999.

[3] Tomoyuki Nishita, Yoshinori Dobashi, Kazufumi Kaneda and Hideo Yamashita. *Display method of the sky color taking into account multiple scattering*. In *Proceedings of Pacific Graphics 1996*, pages 117-132, August 1996.

Illustration 5: Full day sequence

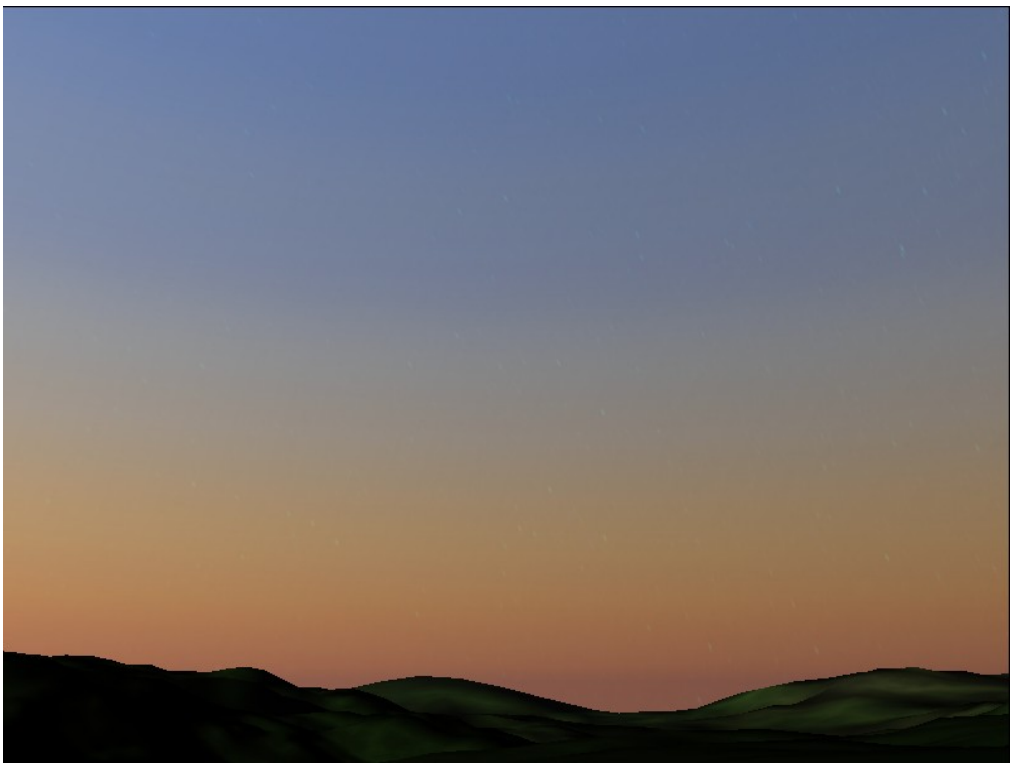

Illustration 6: Light absorption

Illustration 7: Sun glow

Illustration 8: Sky translucency

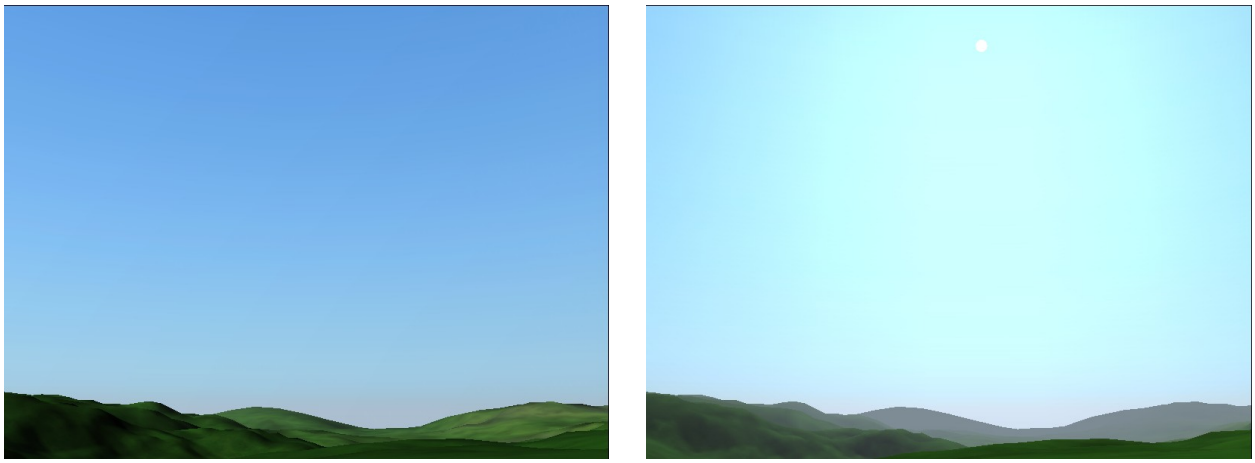

Illustration 9: Dynamic fog