

The Future of Virtual Globes

The Interactive Ray-Traced Digital Earth

M. Christen^a

^a FHNW, Institute of Geomatics Engineering, 4132 Muttenz, Switzerland - martin.christen@fhnw.ch

KEY WORDS: Visualization, 3D data structure, Aerial Photogrammetry, Data Integration, Geo-Information Market, Virtual Globes, Real Time Ray-Tracing, Virtual Reality, Spatial Acceleration, Image Generation

ABSTRACT:

In the past ten years graphics cards have been drastically improved. Being pushed by the multi-billion dollar computer game industry the capabilities and performance of graphics processors (GPUs) increased to support more and more geometric complexity and visual beauty in games.

Today, virtual globes use similar techniques known from 3d-game programming, even though virtual globes use much more image data – which can be in the Petabyte range. Despite these architectural differences virtual globes are – like games - relying on a fast graphics card to maintain a high frame rate of 60 and more images per second at a high screen resolution.

In the past decade there has also been an increase in CPU performance. With the advent of multi-core CPUs and algorithmic improvements it is now possible to reach acceptable interactive frame rates in ray-tracing using high-end multi-core PCs. Ray-tracing calculates an image by simulating the correct transport of light, obeying optical laws and energy conservation. Therefore ray-tracing allows for a substantial quality increase by supporting correct shadow calculation, depth-of-field, caustics, reflections, refractions, and so on. Global illumination adds much more realism to images. The problem of ray tracing is that a lot of computing power is required to create an image, especially for testing large amounts of intersections of light rays with the worlds geometry. There are ways to reduce computing power by using spatial acceleration structures. In the recent years those algorithms have been improved, so that interactive ray-tracing is now feasible.

This paper shows how an interactively ray-traced virtual globe works. It shows how data preparation can be done, including acceleration structures for quick identification of spatial regions. A first implementation of such a physically correct digital earth is shown. Some possibilities with this system are presented and how the future of virtual globes looks like when this technology evolves.

1. INTRODUCTION

1.1 Motivation

Virtual globes use aerial imagery and elevation data to produce 3d-images at any position and orientation on and above the earth surface.



Image 1: Comparing a photo to Google Earth virtual globe

If an image from Google Earth is compared to a photo from the same viewpoint and camera properties, many trivial observations can be made, such as:

- The view highly depends on the daytime, season and weather. For example shadows change over the day or in summer there may be less snow.
- Reflections and flow of water is not visualised in current generation virtual globes

- The photo has a much higher resolution and shows much more details.
- Elevation data looks pretty realistic in the computer generated image. It seems that the main problem lies in low quality image data.

At many parts of the world image data at resolutions of 10 cm per pixel already exists. Of course with time and money better aerial imagery can be created. If there would be an image with 1mm per pixel resolution, streaming this data over the internet is not very realistic: if 50 tiles are required to display an area with 10 cm resolution, more than 800000 Tiles would be required to cover the same area with 1 mm resolution. And if a tile has a size of 256x256 pixel, that would be around 150 GB image data (uncompressed) considering that the same area in 10cm resolution only requires around 9 MB (uncompressed). In addition, graphics cards only hold a limited amount of texture memory at a time, usually between 32 MB and 256 MB in typical "consumer" PCs.

Even if data is available at 1 mm resolution and there is enough texture memory available, the problem with wrong shadows, seasonal changes etc. wouldn't be solved yet.

Another approach must be found to display every spot on the virtual globe in a photorealistic manner.

1.2 Previous work

At the institute of Geomatics Engineering at the University of Applied Sciences Northwestern Switzerland we develop a virtual globe called i3D. It is comparable to Google Earth, NASA World wind and other virtual globe technology available today. These virtual globes use rasterization based real time rendering, which can be accelerated by the GPU (Graphics Processing Unit).

Image data (for example orthophotos) is pre-processed into an image pyramid of different resolutions so that streaming data from the internet is fast enough. The same happens for elevation data and other data channels.

A GPU based real time ray-tracer has been developed (Christen, 2005). It uploads all geometry to the graphics card and traverses a spatial acceleration structure. This ray-tracer has been implemented in the OpenGL Shading Language. On sophisticated graphics hardware this ray tracer runs at interactive frame rates. This ray tracer has been enhanced to also support multi-core.

1.3 Related Work

There are some terrain visualization packages available which ray-trace realistic looking terrains. Those terrains are – in most cases – not using real world data and orthophotos, instead they all use procedural textures which approximate a certain realism, but it remains fantasy. Also, those packages do not support real time rendering. It may take several minutes to calculate one image. One such package worth mentioning is "Terragen", which is capable of photorealistic results for landscape visualization. (<http://www.planetside.co.uk/terrigen/>)

Version 2 is capable to render entire planets and has been used in a science fiction movie ("Star Trek: Nemesis") to generate a realistic looking planet.

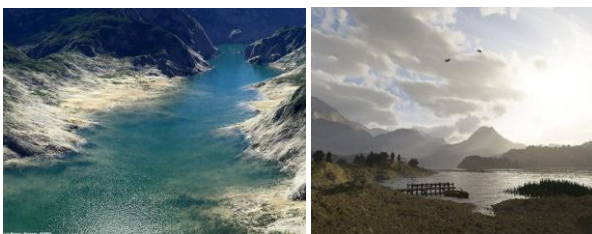


Image 2: Computer generated images using "Terragen"

Also, there are many computer games using realistic looking terrains. In most cases geographic data is not used and artists create fantasy designed data or fractal generated data to generate their "levels". Those games run of course at interactive frame rates. If a good CPU and GPU is used, 60 or even more frames per second can be achieved.



Image 3: Real Time Rendering in the game "Crysis" by Crytek (Released November 2007)

The game "Crysis" uses latest generation of graphics cards to render terrain and 3d-objects as realistic as possible. This kind of rendering is based around a level (or area) and doesn't cover the whole earth. Loading such an area may take a few minutes.



Image 4: Photo (left) and corresponding real time output (right) in the game "Crysis" (courtesy Crytek)

Scenes in games are always created by artists and not by using an automated process. Creating such scenes may take several months.

Many ideas and concepts for today's real-time raytracing has been presented by Wald (Wald, 2004). His concept has been used for the implementation of Jacco Bikers Real Time Raytracer (Bikker, 2007) and Havran (Havran, 2001) who recommends using a kd-tree for ray-tracing static scenes.

2. USING VIRTUAL GLOBES

It is very important to distinguish the usage type of virtual globes. There are many different ways how virtual globes are used today. For example if a virtual globe is used in flight simulation it may be important to have a simulation of clouds and other weather visualization. On the other side if the virtual globe is used for a global or local search, clouds and rain could be very annoying.

2.1 Virtual Globes for Flight Simulation and Computer Games

Virtual globes used in flight simulation must sustain a high frame rate (for example constant 60 frames per second) to avoid simulation sickness and a high image quality. For flight simulation and many computer games it is not necessary to fly from a location to another in very high speed - virtual globes of this type usually have a navigation using realistic aircraft speed which makes it impossible to navigate from New York to Beijing in 1 second. Because of this slow navigation it is possible to add much more details to the current image. Flight prediction allows it to preload images and 3d-objects which are possibly required in near future.

Simulation of the condition of the earth – like climate change – is not included in this type of virtual globe.

2.2 Virtual Globes for Information Retrieval and Deposit

Virtual globes used for information retrieval (including GIS functionality) must provide a high data accuracy. It must allow data representation of raster data, vector data and voxel data. A virtual globe for information retrieval must also allow capturing data. This includes adding new areal imagery, new elevation data, new city models and so on.

Adding the dimension of time may also be necessary. The condition of the surface of the earth – including buildings, elevation and even the atmosphere changes over time may be necessary to visualize.

Another important feature for such virtual globes is spatial analysis.

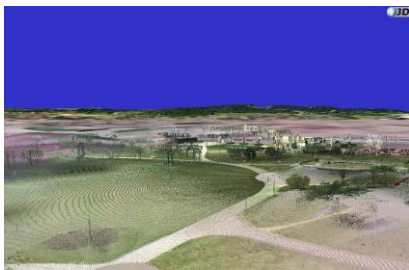


Image 5: Visualization of Voxel Data (Point Cloud) on the virtual globe i3D



Image 6: Visualization Points of Interest on the virtual globe i3D

3. INTERACTIVE RENDERING VIRTUAL GLOBES

3.1 Introduction - Rasterization and Ray Tracing

Rasterization is the most popular approach for generating real-time 3d-images. Rasterization iterates over all geometric primitives - usually triangles - and projects them onto two-dimensional screen coordinates. Before writing it to the frame buffer, its depth value is compared against previously stored depth values (Catmull, 1974).

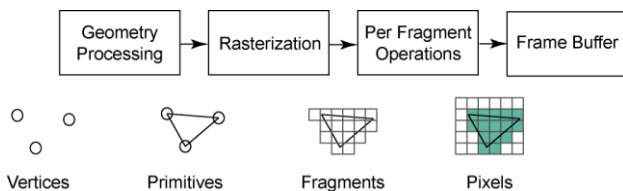


Image 7: Simplified Rasterization Pipeline

During rasterization single primitives are drawn. A relation to other primitives does not exist. Rasterization only processes one primitive (triangle) at a time. Therefore casting shadows from one triangle to the other or calculating indirect illumination is not possible. These effects can only be done using approximations and special tricks.

In addition, the render time of rasterization is linear to the number of triangles.

Ray-tracing is a method of generating realistic images, in which the paths of individual rays of light are followed from the viewer to their points of origin.

The core concept of any kind of any ray-tracing algorithm is to find intersections of a ray with a scene consisting of a set of geometric primitives. The scene to be rendered consists of a list of geometric primitives, which are usually geometric shapes such as polygons, spheres, cones, or any other kind of object as long as it is possible to compute an intersection between a ray and the primitive.

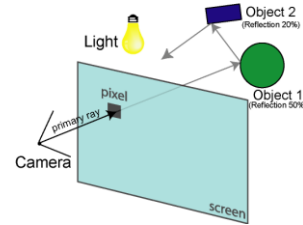


Image 8: Ray Tracing

Ray-tracing has the ability to efficiently render huge datasets. For every pixel a primary ray is cast from the camera to the 3d dataset and tested for intersection. If an object is hit, the ray is traced back including all geometry available in scene. Therefore shadows and reflections are calculated correctly through the whole scene. Radiometric and photometric properties are supported in ray tracing.

3.2 Ray-Tracing a Virtual Globe

Real Time ray-tracing a virtual globe has many advantages. For one, much more image data can be used, because of a much better memory management without graphic cards. Terabytes of data can be used to render one single frame. City models can have much more details because there is no limit of texture memory. Because ray-tracing handles much more data than rasterization, different streaming approaches must be investigated.

3.2.1 Server Side Rendering: Because ray-tracing is a highly parallelizable algorithm, it is possible to have a server or a cluster of servers calculate images and send the result to the clients. While this scenario is highly scalable it requires much computing power for several users. However, the advantage of this method is that not small image tiles are streamed over the internet, instead images in highest quality arrive to the client, regardless of the hardware of the clients system. Sending 60 images of the dimension 1024x768 per second (~47 million pixels per second) can be compressed to a data stream of around 5 to 10 MB per second. A big advantage of this method is that low spec computers can be used to display high resolution photorealistic animations, while the datacenter computes the images. The disadvantage is that a fast network connection to the data center must be available all the time. Another disadvantage is of course the high cost for such a data center.

3.2.2 Client Side Rendering: Client side ray-tracing a virtual globe requires a fast system with at least 8 CPU cores of the latest generation - like Intel Penryn - to maintain more than 30 frames per second. The problem with client side rendering is that tile data must be streamed from the data center to the client. Furthermore, client side rendering can be categorized into two cases:

3.2.3 Data over Network: Data is being streamed over a network - usually the internet. This is well known procedure already used in virtual globes like Google Earth, where small data tiles are downloaded. Usually you have to wait a few seconds until the highest resolution is available, especially when navigation speed is fast or the internet connection is slow. Like rasterization based virtual globes, this is still the preferred way to stream data.

3.2.4 Local Data: Reading tile data from hard disk is much faster than over a network and results in much better quality in much less time. However, this requires every user of the scene to have Gigabytes of tile data on their hard drives. Instead of hard disks, memory sticks or solid state disks can be used because of their smaller latency to access files.

3.3 Data Preparation

Image Data in current generation virtual globes is based on the original data provided to the virtual globe. Usually this data is imagery in the human visible light spectra. Ideally, the aerial imagery and the image visible on the computer generated virtual globe looks the same. However areal imagery has properties like shadows that shouldn't be part of the image. Removing shadows may be required to enhance realism.

3.3.1 Procedural Images: Procedural texturing is a well known method (Peachy,1985, Perlin, 1985) which has been used in many applications. A procedural image is an image created using an implicit function. There are many ways to create such functions like stochastic patterns, pattern based images (Lefebvre, 2003). A way to allow one to one mapping from a 3D objects surface to its texture space has also been presented (Carr, 2002). Creating such imagery usually involves using a noise function which is repeatable, has a known codomain and is band limited (Perlin, 2002). Unlike pixel based images, procedural textures do not have a limitation in resolution.



Image 9: Example Procedural Textures

3.3.2 Creating Procedural Images from real world data:

The basic idea is to use standard classification methods to calculate an appropriate procedural image from a catalog of different procedural materials and apply it to the correct location. The more materials and classifications are available, the better is the result of final procedural material. Combined with additional vector data for streets, road markings, ground plans and so on, it is possible to achieve a very high image quality.

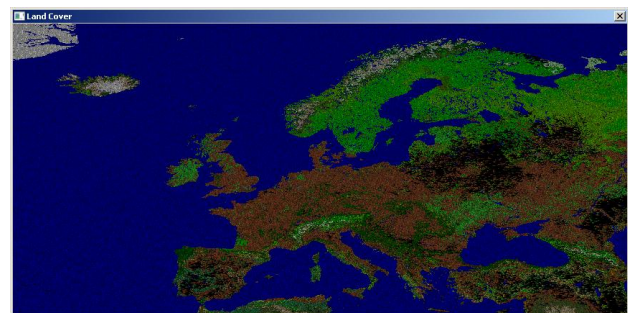


Image 10: Land Cover Data over Europe with Procedural Interpretation

3.3.3 Combining Areal Imagery with Procedural Imagery:

Areal Imagery is often preferred to land cover based procedural visualizations because of the increased level of realism. Some parts of the areal imagery can be enhanced using procedural image generation - for example streets, roads, rivers, shores, lakes, and so on. The result would be a mixed areal image tile with high resolution parts of generalized data built with procedural algorithms.

There are already a number of algorithms available to create realistic looking surfaces. For example rendering realistic looking clouds - also useable for weather simulation - can be implemented using the model presented by Harris (Harris, 2001).



Image 11: Cloud rendering with anisotropic Rendering (Harris, 2001)

Another example is simulating ocean water, which has been shown in many works. A realistic looking ocean was presented by Tessendorf (Tessendorf, 2004).



Image 12: Ocean Water simulated using the model of Tessendorf, 2004

Creating generic, realistic looking procedural imagery for every material on earth is not a trivial task. Over time such models can be improved and adapted and accelerated for real time ray-tracing.

3.4 3D Objects

3.4.1 Vegetation: Vegetation Rendering is another important part of a virtual globe to increase visual realism. There are some commercial 3d-tree model libraries available like "SpeedTree" or "XFrog" with numerous of tree species. On the virtual globe trees must be put on the surface according to raster based land use data or land coverage vector files. It would be possible to approximate the shape of the original tree, but the work to do this for every plant on the world would be enormous. It makes more sense to use an area based wood classification and represent that area with species and approximated density of trees available in that location. If an area of interest requires detailed tree representation this can still be manually done after the approximated tree layout has been set.



Image 13: Tree Rendering on the virtual globe using the "SpeedTree" library.

3.4.2 City Models: Procedural 2d-reconstruction of buildings on the earth surface makes little sense - ideally, a building shouldn't be part of the imagery, a city should only be available as 3D model. Visualization of roads including markings can be created from appropriate vector data as described in 3.3.3.

3.5 Implementation of a Real-Time Ray-tracer for virtual globes

When ray-tracing a huge number of rays it is not efficient to intersect each ray with every triangle of the underlying elevation model on the virtual globe. An acceleration structure must be used to reduce the number of total intersections.

Many spatial data structures can be used for ray-tracing. Partitioning space using a kd-tree based on the surface area heuristic has proved to be a good way for static geometry data (Havran, 2001).

This kd-tree can be adapted to be used for all geometry (elevation data) on the earth ellipsoid.

Because image and elevation data of the earth can exceed petabytes of data, a level of detail approach has to be used to allow streaming geometry and image information over a network. The key concept of level of detail is the exploitation of the fact that humans are unable to notice details in objects that he isn't attending. (Mack, 1998). Over time the best resolution of the observers area is downloaded and the image generated uses the best available resolution of both image and elevation.

For the virtual globe a streamable kd-tree has been developed in which each node contains a reference to low resolution geometry and information where to download higher, detailed geometry and imagery of the same area. This way low resolution data is always available for the whole planet and detailed data visible by the observer can be streamed over the network.

The main bottleneck when real time ray-tracing the virtual globe is traversing the acceleration structure and shading using procedural and downloaded imagery. Downloading new data itself is not a bottleneck for the image generation, it is a bottleneck for visualization speed, as already known from rasterization based virtual globes.

Traversing the kd-tree for intersection tests must be optimized. This can be done by traversing several rays at once using SIMD instructions (Wald, 2004). Using larger ray packets and using multicore processors among other tricks can speed up the ray-tracer even more. (Reshetov, 2005).

To achieve interactive frame rates when rendering a virtual globe using real-time ray tracing, the minimum specification of is a dual quad core system (total 8 cores, 3 GHz Intel Xeon 54xx) is required. Such a system is not cheap and not available for everyone. In a few years such systems may be wider available on the mass market.

4. CONCLUSIONS

The need to improve the visual quality of current generation virtual globes is around. Using ray tracing allows to break the limited maximal geometry and textures of rasterization based

virtual globes. Using real time ray-tracing for virtual globes is a very cpu intense task. Streaming data requires high speed data transfer for best image quality. When realistic looking visualization is required - for example for broadcast applications, then using ray tracing for virtual globe rendering is highly recommended. For low resolution information retrieval, like global search where visualization itself is not the most important part, rasterization based virtual globes are still a good choice. A ray-traced virtual globe requires hardware that is - today - too expensive for the normal market. In a few years from now when multiple quad core processors are cheap enough for the mass market this kind of rendering can be used on common mass market computers to create realistic looking visualizations of every spot on the earth.

5. REFERENCES

- Appel, A. 1968. Some techniques for shading machine renderings of solids. In Proceedings of the AFIPS Spring Joint Computer Conference. Vol. 32. 37–45.
- Bikker, J. 2007, Real-time Ray Tracing through the Eyes of a Game Developer, IGAD / NHTV University of Applied Sciences
- Carr N., Hart J, 2002, Meshed atlases for real-time procedural solid texturing
- Catmull, E., 1974, A subdivision algorithm for computer display of curved surfaces. Ph.D. dissertation. Department of Computer Science, University of Utah, Salt Lake City, UT.
- Christen, M., 2005, Implementing Ray Tracing on the GPU, ShaderX4 Advanced Rendering Techniques, Charles River Media, pp. 409-427
- Dachsbacher, C., 2006, Interactive Terrain Rendering: Towards Realism with Procedural Models and Graphics Hardware
- Hurley, J., 2005, Ray Tracing Goes Mainstream, Intel Technology Journal, Compute-Intensive, Highly Parallel Applications and Uses, Volume 09, Issue 02.
- Harris, M. J., Lastra, 2001, A. Real-Time Cloud Rendering. Tech. rep., University of North Carolina at Chapel Hill.
- Havran, V., 2001, Heuristic Ray Shooting Algorithms. PhD thesis, Czech Technical University, Praha, Czech Republic
- Iglesias A., 2004, Computer graphics for water modeling and rendering: a survey, Future Generation Computer Systems, v.20 n.8, p.1355-1374, November 2004
- Lefebvre S., Neyret F., 2003, Pattern based procedural textures, Proceedings of the 2003 symposium on Interactive 3D graphics, April 27-30, 2003, Monterey, California
- Mack A., Rock I., 1998, Inattentive Blindness. MIT Press
- Peachey, D. R., 1985. Solid texturing of complex surfaces. In Proceedings of SIGGRAPH 85. 19, 3 (July), 279–286.
- Perlin, K., 1985. An image synthesizer. In Proceedings of SIGGRAPH 85. 19, 3 (July), 287–296
- Perlin, K., 2002 Improving noise. In SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, pp. 681–682.
- Reshetov, A., Soupikov, A., Hurley, J., 2005, Multi-Level Ray Tracing Algorithm. ACM Transactions on Graphics.
- Tessendorf, J., 2004, Simulating Ocean Water. In SIGGRAPH 2004 Course Notes.
- Wald, I., 2004, Realtime Ray Tracing and Interactive Global Illumination. PhD thesis, Saarland University.
- Whitted, T. 1980. An improved illumination model for shaded display. Commun. ACM 23, 6 (June), 343–349.