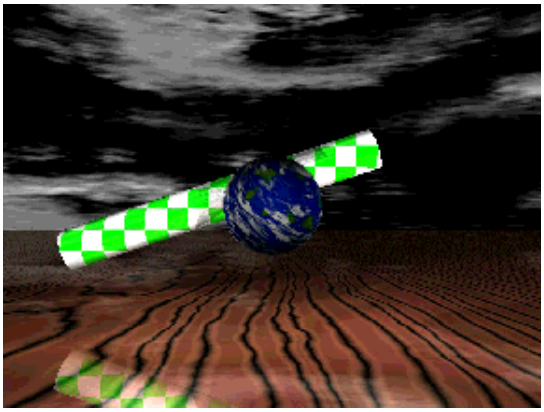
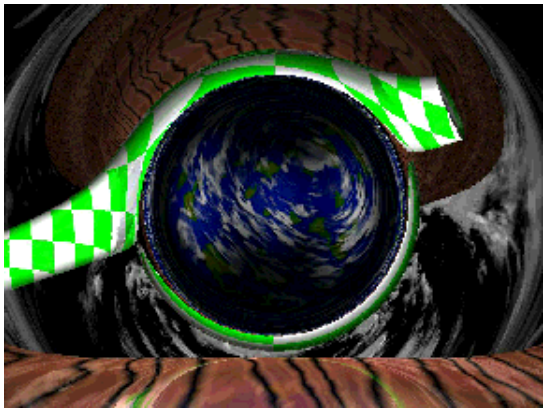
A distributed ray tracer for static black hole animations





]

Tzer H

[Abstract | Introduction | Goals | Schedule | Achievements | Individual Contributions]

[Lessons Learned | Acknowledgements | Bibliography | Appendix]

Abstract

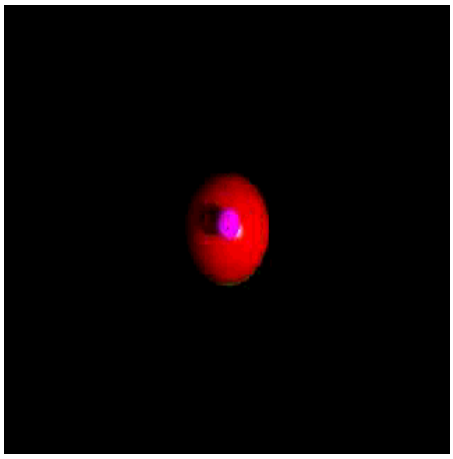
The goal of this project was to accurately simulate what a static scene containing a black hole would look like. The calculations involving a black hole are complex, and the appearance of a scene involving one is bizarre. For this project, we shall only concentrate on visually explaining the distortion effects of a single non-rotating, non-charged, non-emitting, stationary black hole, since our goal is to explain to a novice the effects a black hole has, and it would be more difficult to comprehend with extra parameters. Yet another simplification we made for the animation to be of a static scene. Thus, an observer can play god and fly through space with everything around him stationary, enabling him to fully concentrate on the ray-bending effect of the black hole.

This project was created as the final project for 6.837 Computer Graphics at MIT in the fall of 1999. The three members involved are Alex Yip, Brian Lin and Tzer Hung Low.

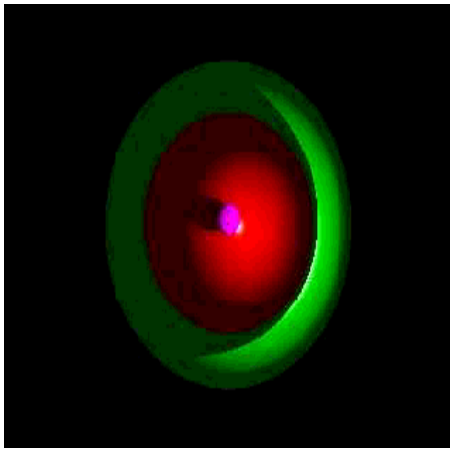
[Back to Top]

Introduction

Loosely speaking, a black hole is a region of space that has so much mass concentrated in it that there is no way for a nearby object to escape its gravitational pull. It is impossible for a human to travel very near a black hole, which has a mass like that of the Sun. Indeed, the high gravity itself would likely pose insurmountable problems.



For the sake of illustration, suppose you got into a spaceship and traveled straight towards a black hole. Starting from a long way away from the black hole, you just turn off your rockets and coast in. At this stage, you would not feel any gravitational forces at all. Since you would be in free fall, every part of your body and your spaceship would be pulled in the same way, and so you feel weightless. As you get closer and closer to the center of the hole, however, you would start to feel "tidal" gravitational forces. This implies that if your feet were closer to the center of the black hole than your head, then the gravitational pull on your feet would be stronger than that on your head. This asserts a stretching effect as though you were on a rack! These tidal forces get more and more intense as you get closer to the center, and eventually they will rip you apart.



Since none of us would ever be able to view the effects of the black hole in realty, we have to resort to computer images. However, it is very difficult for us to visualize the effects of a black hole on perceived scenes, because the idea of light bending violates our intuition of the world that we live in. For example, if we had a sphere centered around a black hole, with radius greater than the Schwatzchild radius, then the observer should be able to see the entire surface of the sphere from any point of view. Another interesting but counter-intuitive example is as follows. If the observer is near the surface of the sphere, and looked tangent to the sphere, then one would not only be able to see the back of his head and the surface of the sphere, but may see it multiple times. In fact, these multiple images can appear even if the observer is not looking on the horizon. Lastly, if there were a table near a black hole, an observer would be able to see both the top surface as well as the bottom surface simultaneously.

Therefore, scenes like these are very difficult to understand, even if we had a still image. An animation could better illustrate the nature of bent light and black holes.

[Back to Top]

Goals

Since we are still a long way off from physically viewing a scene near a black hole, this barrier offers the primary motivation for us to render an animation of such a scene through ray tracing. Certainly, this is the best we can ever do for some time. There are several other motivations for this project. We wanted to experiment with new methods of ray tracing for rays that do not travel in a straight line. This presents a hard problem, and we chose to solve this by approximating the ray path by many successive rays-segments, optimizing the calculations by increasing the ray-segment length for regions that are further from the black hole. We also had to consider super-sampling for regions where rays of light may diverge due to the black hole.

We also wanted to indulge in a particularly difficult and lengthy animation. This project is certainly a candidate, since each ray cast is divided into many smaller segments and therefore far more tedious. We solved the problem of the long time taken to render the animation by creating a distributed system where a central server load-balances with any number of clients, each of which can leave or enter the project at any time. Last but not least, we are interested in mathematically understanding and expressing the black hole equations into a computer program. This involves ensuring accuracy and efficiency, such as by using the Runge Kutta method.

[Back to Top]

Schedule

11/8/1999

- Complete the distributed framework for animations

11/15/1999

- Complete survey on Black Holes and Space-time curvature using General Relativity

11/18/1999

- Start code on Black Holes:
- First only consider non-rotating, stationary black holes, with no regards to special relativity

11/20/1999

- Finish Code to integrate Mass into the ray tracer.

11/24/1999

- Start animation. Animation will consist of a scene taken from somewhere, or that we choose to render ourselves, with one or more black holes in it and which we fly around the black hole, and perhaps inside.
- One group may do animation while another continues to enhance the ray tracer. All file formats and conventions will have been decided by then.

11/26/1999

- Optimize the ray tracer.
- Correct some errors due to boundary conditions
- Attempt to further increase the accuracy. For example, change from the use of floats to doubles.
- Ensure that there are no "specks" in the images due to highly diverging rays.

12/1/1999

- Finish the final project write-up
- Render a few high quality snapshots for the final project write-up.

[Back to Top]

Achievements

Understanding the black hole equations and the right metrics to use

Understanding the black hole equations and the right metrics to use

Our first big step was to understand the correct equations and figure out which equations are more suited for our purpose according to what metric we are using to define distance in our space. It is hard to say what one would actually see since we are trying to project four-dimensional space onto two dimensions, especially since the space cannot be defined by normal Cartesian coordinates and there are discontinuities at the event horizon of the black hole. This is particularly a problem with the Schwartzchild metric. However, it is the simplest coordinates to use for non-rotating black holes. Since the Schwartzchild radius is defined by the distance from the center of the black hole for which the circumference is 2pi*R, it is actually convenient for us to calculate the path of the ray of light for a space that we expect to have. That is, a space in which the spatial curvature is flat. If we pretend that the radial distance to successive radii is the difference between the two.

In fact we came upon many paradoxes and issues regarding the black hole equations not only in the

beginning but as we continued to write the code. This led us to think more deeply about how ray tracing was done.

Here are several problems and issues we came along with.

- Light rays are no longer reversible. That is, a light ray going towards the black hole does not travel the same exact path as a light ray coming out of the black hole or rather one that never escapes. This gave us much headaches for a long time, proving to ourselves that backward ray tracing is still valid. We have trying to calculate the ray backward in time and even considered forward ray tracing for a while. However, we figured out that as long as we appropriately check for discontinuities and irregularities near the black hole horizon, which is at 2M, and at the photon orbit, which is at 3M. In addition we also needed to check for energy inflection points. In the end, we proved to ourselves that as long as we do not go inside the black hole, backward ray tracing still holds.

- Is the coordinate warped or the space warped. Many times during our project as bugs came at us, we often rethink what coordinates actually mean. Now, if space is warped, is that the same as warping the coordinates? Even now we are unsure. What does distance mean? And who is measuring the distance, and by what metric. How does one convert a coordinate system where the distance that one travels between two points vary depending on which point you are on to a Cartesian coordinate system. Especially when the distance between two points can be not only infinite but also imaginary. What we do now is render the space by converting a point in warped space to real space and pretending that the objects don't move. But we say nothing about the distance between the objects. This seems to suffice, and is what we have seen others do. But whether it is truly correct, is a difficult question. Converting back to Cartesian coordinates in flat space seems logical. But in the end, is that REALLY what you see? We believe so, but no one is really sure.

- The affine parameter, Lambda, blows up problems. Here is a problem to which we do not know if there is an easy solution. This lambda parameter determines the size of the step dr, and dphi. The problem is that there is no lambda that you can pick that can could satisfy all or even two of these requirements at the same time, that dphi or dr does not overflow, that dphi or dr is not zero incorrectly, that the step size is big enough so that it wouldn't take forever to render, and that the step is accurate. Here is why, as r gets small dphi becomes infinite which would lead you to decrease lambda by so much that it would take an hour to render one ray, since the change in r would be nearly zero, if not in fact zero from rounding if b is big. If b is small, or sometimes even zero, dr is infinite regardless of lambda which causes many tunneling problems. If again we make lambda small the tracing gets much slower. It is more annoying when we discover that when b is not small and you can make the approximations and have big steps, but we could still be forced to make lambda small to avoid overflow. Also, if you are far from the black hole you can also

approximate the ray as straight, but this is when dr and dphi become invalid. So you can't make lambda big. It is all a huge very complicated circle to which there seems to be no solution. We have MANY MANY functions that dynamically change lambda and other parameters in order to trace correctly and complete the tracing in a reasonable time. They are fairly complex and we will not go over them here. But they involve taking advantage of every bit of information we have. This also makes hard for debugging since one wrong function in the chain of functions that tweak lambda incorrectly and we may not even see an image.

- We also try to solve for normals, shadows, and effects for the new curved ray. Initially, we wanted to edit POVray so that we would be able to take any image and render it with a black hole in it, or at least that is what we thought was possible. However, there are a couple unsolvable problems. One is, there is no way to calculate the normal of a ray at a certain point, given its starting point and ending point. In fact, although there is a unique ray that goes through these two points, there is no way to calculate the path of this ray. One would just have to try ALL the possible directions that the ray can start at and see if it happens to reach that point, a ridiculous task. Thus, normals and shadows are practically unsolvable unless you do a few approximations or tricks.

- Matrix conventions, POVray conventions. It took us a while to learn the several conventions that POVray uses in terms of coordinates, and normalizing vectors, and matrices. Especially since they are not all entirely consistent throughout the code. This caused us to waste a lot of time trying to fix something else when their matrix indexing was reversed from what they had used previously in another function.

Tracing Algorithm

In our simulation, we assumed that there is only one black hole, and that short line segments can approximate rays of light. Now we describe this method of approximation.

First, we translate the black hole center to the origin, then do a basis change to map the black hole and the initial ray of light from the eye to the Z-plane. In other words, we map the scene so that the Z-plane contains both the line segment and the center of the black hole. The light ray cannot leave that plane, because there is no net force pushing or pulling it off that Z-plane. This allows us to simplify our calculations to two dimensions, on the Z-plane.

Second, we transform the coordinates in the Z-plane to polar coordinates. Using polar coordinates to describe the geometry is more appropriate since the distance of the light ray from the origin where the black hole is located is referred to many times. A pair of differential equations, parameterized by parameter lambda describes the curved light path in this plane. They are:

- dphi/dlambda = 1/r^2

- (dr/dlambda)^2 = 1/b^2 - (1/r^2) * (1-2m/r)
- c = G = 1
- b = L/E.
- Where r and phi describe a point in the Schwartzchild metric.
- Where c is the speed of light, G is the gravitational constant.
- Where b is the impact parameter, L is the angular momentum and E is the energy of the ray.

To solve for the path of the ray, we use the Runge-Kutta Method, to approximate a solution to the differential equations to minimize error. There are several reasons why we chose this over the Euler?s method.

1. The Euler method is not very accurate, especially when a build-up of error occurs with more and more ray-segments that are cast to approximate the ray path.
2. It is not very stable.

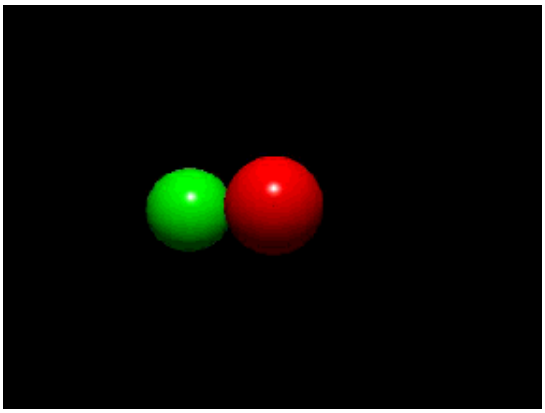
After we calculate dr and dphi, we have to perform numerous calculations to ensure these are valid numbers and accurate. These tests include testing for tunneling through energy thresholds, testing if the ray is going towards the black hole too fast, testing if the ray has too much or too little curvature, testing to see if the ray is inside the black hole and testing to see if we are outside an effect radius at which point we cast a straight ray segment to infinity.

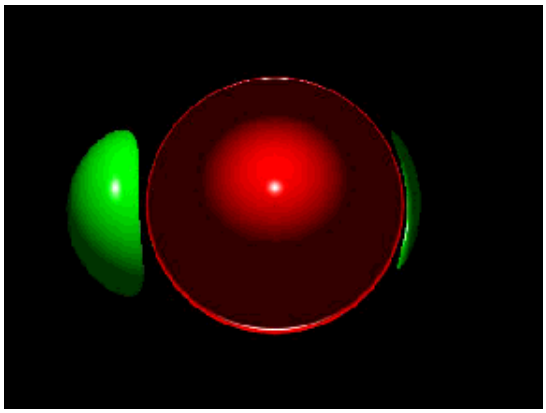
Then we tweak dr and dphi based on b, r, and M.

From here on, we can use the standard calculations for finding intersection, reflection, and transmission. After finding the intersection, reflection and transmission, we create a new ray and continue the process anew, just as if this were a new eye ray.

In the end we were fairly capable of rendering primitive objects with texture reflections and most operations that depend on casting new rays. Here are a few examples:

Left: W



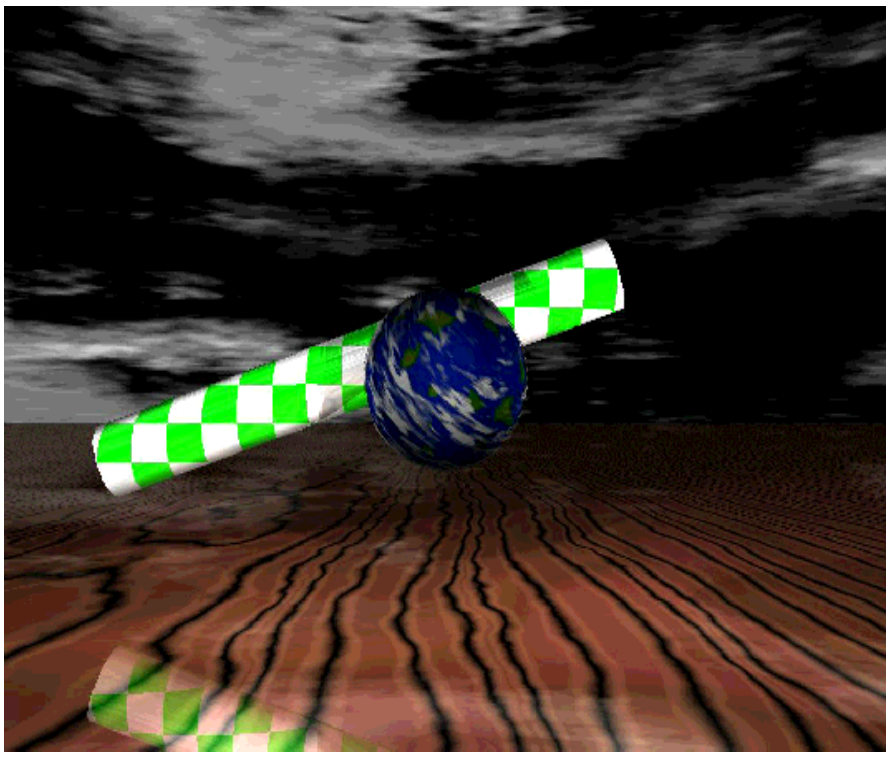


Left: W
a black

Right: V
black h

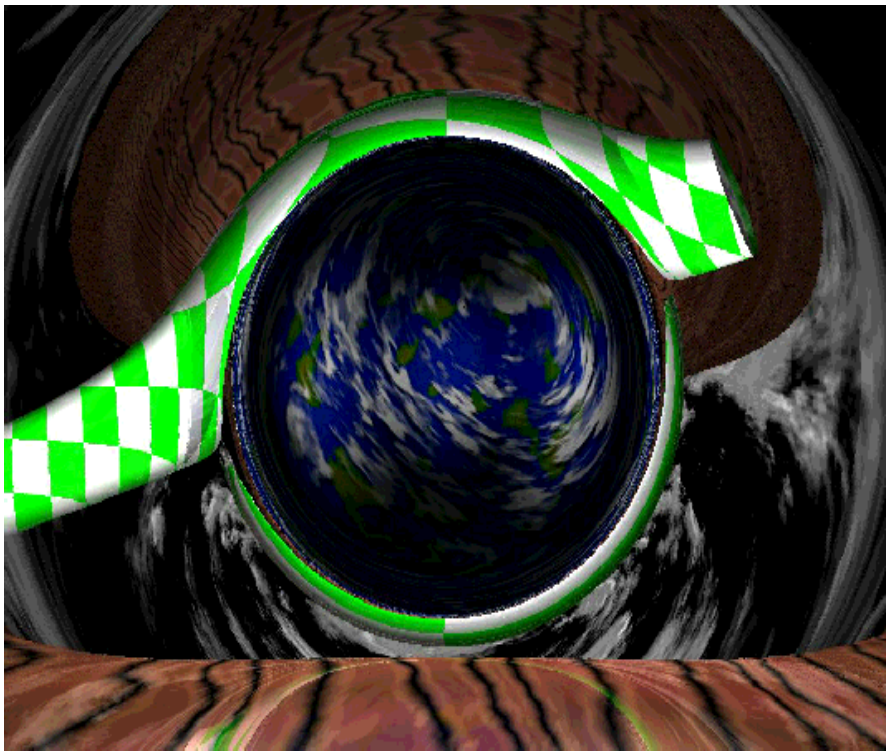
Here we compare our rendering to the only other ray tracer we know of that does something similar. Ours is very similar, and in fact more correct then theirs. Since they allow you to see things inside the Schwartzchild radius of the black hole, which is incorrect. We also believe from experience that that is the correct curvature in all places. In addition, they also have small flaws here and there, which although we may have at certain times are less significant than theirs are. Their is the most impressive ray traced black hole scene we have come across and is the only one that all of the sites which we have come across link to. Here are their two images, and also our own version:

On the right is our ray-traced picture without a black hole.

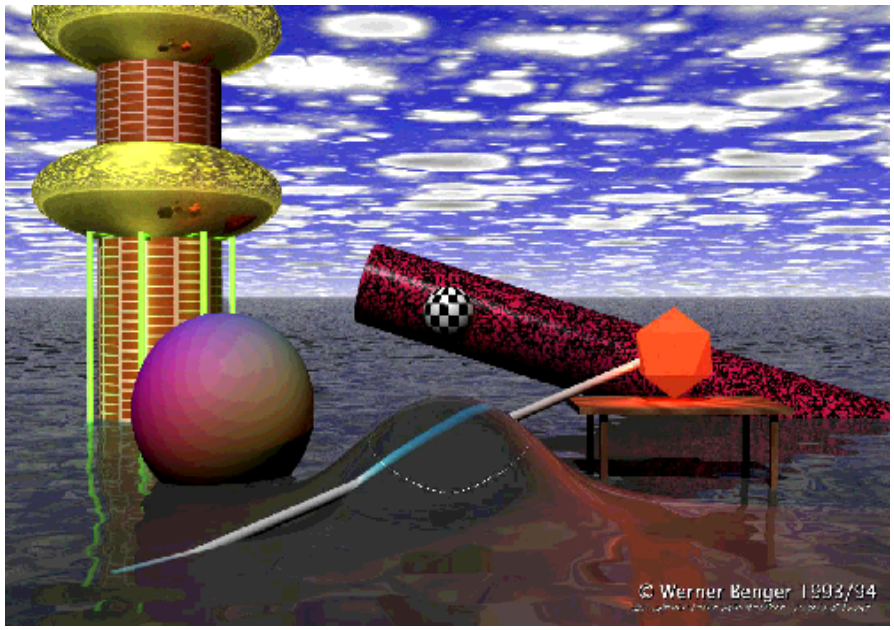


On the right is our ray-traced

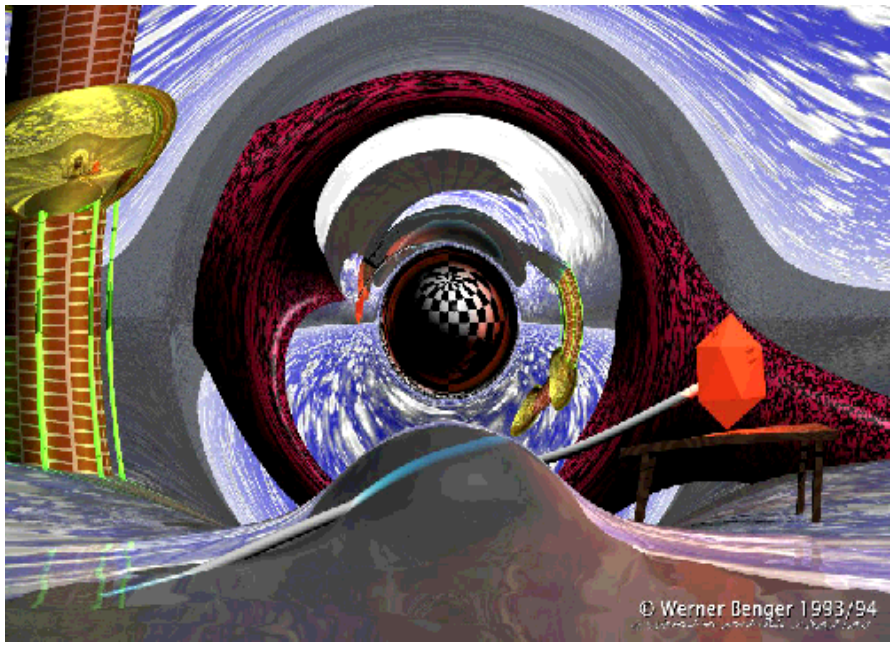
On the right is our ray-traced picture with a black hole.



On the right is our "competitor's" ray- traced picture without a black hole.



On the right is our "competitor's" ray- traced picture with a black hole. Note that you to see things inside the Schwartzchild radius of the black hole. This is incorrect.



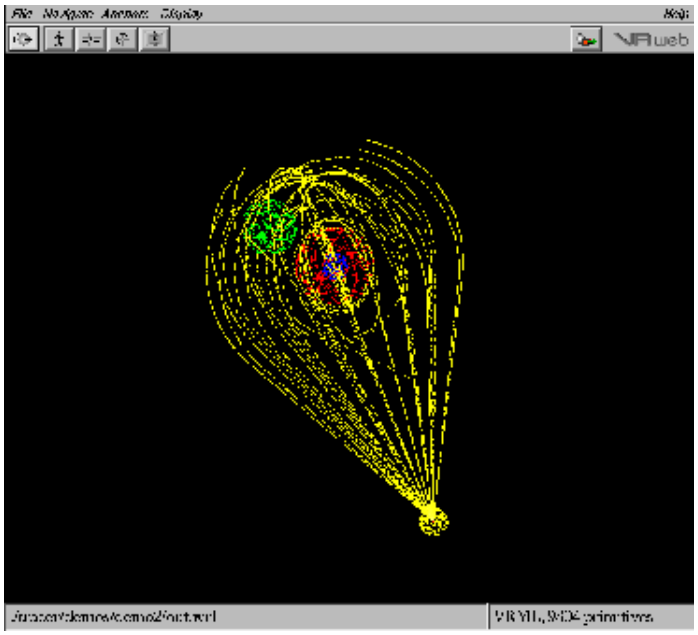
Distributed System

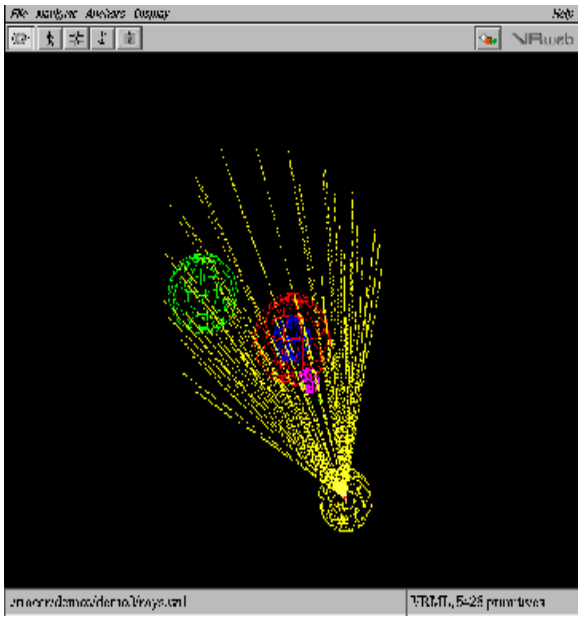
Since we have to generate more ray segments and do much more computation than the standard POVray, we built a system to distribute the work over many workstations. This cuts down the time taken to render the animation. A central server load-balances the workload with any number of clients through the network, each of which can leave or enter the project at any time.

The distribution system is built in Java, and runs as a layer above the POVray tracer. There is one central server that organizes the workload, and distributes jobs to its assisting workstations.

Visual Debugging

We developed a system to output an ivray file that describes the path of the ray cast. An example is shown above, where the eye is represented by the center of the yellow sphere.





Here is another example. On the left above is an image light rays in the presence of the black hole. On the right shows the path of the rays in the same scene but with the black holes removed. As expected, the rays are straight.

[Back to Top]

Individual Contributions

As this project was rather involved, we helped each other with different problems although we divided the work into distinct parts. However, the official work distinctions are hereby listed, in close to chronological order.

- Alex and Tzer Hung are responsible for modifying POVray to be a distributed ray-tracer.
- Brian is responsible for comprehending and explaining the black-hole equations.
- Tzer Hung is responsible for the matrix transformations for mapping each ray to the Z-plane, as well as, the necessary coordinate changes.
- Alex is responsible for creating a new object in POVray, thereby allowing us the framework to add black holes into .pov files.
- Brian and Tzer Hung are responsible for implementing the first rough Euler's approximation to the ray's path.
- Alex is responsible for creating an automated output that shows the paths of the rays that the ray tracer cast. He is also responsible for stepping through the debugger to pick out mistakes, and

errors such as round-off errors or argument-passing errors.
- Tzer Hung is responsible for preparing this write-up, as well as solving certain mathematical equations, such as that of a cubic equation.
- Brian and Alex are responsible for implementing the Runge Kutta method, and correct certain errors associated with the path of the ray.

- Alex is responsible for optimizing rays cast outside the effects of the black hole.
- Brian is responsible for optimizing length of each ray cast, as a function of its curvature.
- Brian and Alex are also responsible for debugging Brian's optimizations.
- Brian and Alex are responsible for preparing & rendering demo images for this write-up.
- Brian and Alex are responsible for adding their perspectives on the write-up.
- Tzer Hung is responsible for merging all parts of the write-up, and posting it to the web.
- Tzer Hung, Brian and Alex will jointly prepare an animation for presentation.

[Back to Top]

Lessons Learned

One of the biggest lessons we learned while developing this extension to POVray, was that discrete approximation is very difficult when dealing with realistic natural models. In our system, the problem rears it's head when certain small times steps turn into huge steps in ray position and direction. These steps can become very large because of singularities in the physical system, such as the rays that travel directly towards the black hole. These rays cause calculations that produce huge values. Although these values make sense in the pure mathematical world, they cause failures in our approximated world. One might think to reduce the size of the time step, but any arbitrary or non-continuous change in time step can produce some tearing in the final image. The problem is that continuous functions for time step are difficult to imagine, because the singularities are related to ray position by complex multivariable expressions. Thus, smoothing out the ray segment length turned out to be quite a challenge.

Another complication that made large steps even more of a problem was variable overflow. This bug was very hard to track down, since the only indication of overflow was the appearance of black pixels in our scenes. This coupled with the large step caused bugs that were quite challenging to remedy.

One of the more obvious lessons we learned was that visualization helps a tremendous amount when trying to understand large amounts of data. During many of our debugging phases, we wanted to observe the path our rays were taking, but all we had was a stream of 3d coordinates. This didn't help us very much, considering visualizing 2d points is difficult enough without the 3rd coordinate. Our solution was to produce a VRML scene during trace time that plots data points of the rays cast. Not only were these

models a great help for our debugging process, but they show how the light is affected by the black hole very clearly.

In its current state, the tracer does not bend the light that comes from light sources because the lighting equations assume that light travels in a straight line. It only takes the dot product of the straight ray direction and the normal of the colored surface. Reverse ray tracing makes it very difficult to calculate the ray that *would* have hit a point, from a light source, because the algorithm would have to somehow figure out the path a ray would take from a source, conditional on a black hole, to hit an intersection surface. This is a very difficult problem.

A simulator that could accurately approximate bent light from the light sources would be a forward ray tracer or a radiosity tracer, that bends the rays as it casts them from the light sources,(color patches). This way, the surfaces get lit with light as if the light were bending. Then, unlike a normal radiosity-shading algorithm, ray trace with depth 1, so that the final rays from the eye are also bend, before they hit the objects in the scene. This would produce a more accurate scene in terms of lighting effects.

[Back to Top]

Acknowledgements

- We would like to give our sincere appreciation to our TA Kari Anne Høier Kjølaas. Without her guidance and suggestions, this project would not have been possible.
- We would also like to thank Prof. Seth Teller for his fascinating lectures, and also his concise lecture notes that proved to be invaluable especially when figuring out matrix transformations.
- We would also appreciate the help of the tutor in Brian’s dorm, who pointed us to use the Runge Kutta method.
- We would to acknowledge the contribution of Annie M Lo, whose spirited discussions about black hole gave us further insights into the problem.

[Back to Top]

Bibliography

- Gear, C.W. 1971, Numerical Initial Value Problems in Ordinary Differential Equations (Englewood Cliffs, NJ: Prentice-Hall).
- Acton, F.S. 1970, Numerical Methods That Work; 1990, corrected edition (Washington: Mathematical Association of America), Chapter 5.

- Stoer, J., and Bulirsch, R. 1980, Introduction to Numerical Analysis (New York: Springer-Verlag), Chapter 7.
- Lambert, J. 1973, Computational Methods in Ordinary Differential Equations (New York: Wiley).
- Lapidus, L., and Seinfeld, J. 1971, Numerical Solution of Ordinary Differential Equations (New York: Academic Press).
- Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathematics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York),
- Gear, C.W. 1971, Numerical Initial Value Problems in Ordinary Differential Equations (Englewood Cliffs, NJ: Prentice-Hall), Chapter 2. [2]
- Shampine, L.F., and Watts, H.A. 1977, in Mathematical Software III, J.R. Rice, ed. (New York: Academic Press), pp. 257-275; 1979, Applied Mathematics and Computation,vol.5, pp. 93-121. [3]
- Rice, J.R. 1983, Numerical Methods, Software, and Analysis (New York: McGraw-Hill), X 9.2.
- A.P. French. Special Relativity, W.W. Norton & Company, 1968.
- Einstein, Albert. Relativity. Three Rivers Press, 1961.
- Schutz, Bernard F. A first course in general relativity. Cambridge University Press, 1996.

[Back to Top]

Appendix

Here are some sources available on the web:

http://www.ncsa.uiuc.edu/Cyberia/NumRel/GenRelativity.html

http://people.whitman.edu/~whitehno/bh/bh.htm

http://jean-luc.ncsa.uiuc.edu/

http://www.astro.ku.dk/~cramer/RelViz/

http://math1.uibk.ac.at/~werner/black-earth/

http://casa.colorado.edu/~ajsh/schw.shtml

http://math1.uibk.ac.at/~werner/light/

http://math1.uibk.ac.at/~werner/bh/gvsim.html

http://vishnu.mth.uct.ac.za/omei/gr/

http://www.astro.queensu.ca/~musgrave/cforce/blackhole.html

[Back to Top]
