Final Project Report: Interstellar Black Hole Rendering

Project Topic

This project focuses on rendering a black hole using OpenGL 4.1 and GLSL shaders. Inspired by the depiction of black holes in the movie Interstellar[4], the goal was to create a realistic visualization (*Figure 1*) that includes gravitational lensing, an accretion disk, and a dynamic skybox environment.

Figure 1: Snapshot of Project Render

Rendering Pipeline

The rendering pipeline (*Figure 2*) for the black hole scene begins with rendering the base scene, followed by a brightness pass that extracts high-intensity areas. The extracted brightness is downsampled to reduce resolution, then upsampled to create a soft blur effect, forming the bloom effect. The processed bloom is then combined with the original scene to enhance bright areas, such as the glowing rings around the black hole. Finally, the tone mapping stage adjusts the final color and brightness by using the *ACES Filmic Tone Mapping Curve*[5] to ensure a visually balanced output.

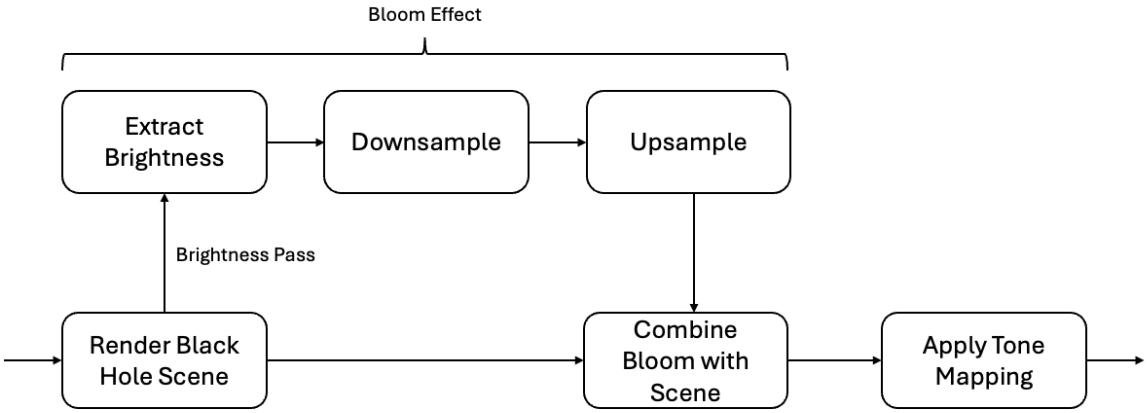

Figure 2: Rendering Pipeline

Shader Implementation of Main Scene

1. Gravitational Lensing (Ray Marching)

Gravitational lensing occurs when light bends due to the intense gravitational field near a black hole's event horizon. This shader simulates this effect through ray marching, where rays of light are marched iteratively to see how they interact with the black hole's gravitational field.

Ray Marching Calculation: Rays are marched from the camera to the black hole. At each step, gravitational acceleration is applied to the ray direction based on the simulated mass of the black hole.

Termination Criteria: The ray march stops once it reaches the black hole's event horizon (distance < 1.0), or after a fixed number of iterations (300 in the code).

Acceleration: The gravitational acceleration at each step is computed using a simplified approximation[3] of general relativity, specifically tailored to simulate the bending of light rays due to gravitational lensing near a black hole:

$$acc = -1.5 \times h^2 \times \frac{pos}{r^5}$$

where h^2 is related to the cross product of the position and ray direction, pos is the current position in space, and r is the radial distance from the black hole.

2. Accretion Disk

The accretion disk is a rotating disk of gas and dust particles that spirals into the black hole. It is visualized with the *adiskColor* function, where various physical parameters control the appearance of the disk.

Density and Particles: The density of the accretion disk decreases with distance from the black hole. The density is also affected by the vertical distance from the disk's center, and particles are distributed using a noise function.

Noise and Particle Motion: Simplex noise[1] is used to simulate the movement of particles within the disk. This adds a level of randomness and realism to the motion, where the noise is animated over time to give the appearance of dynamic movement.

Coloring: Particles in the disk are rendered using a texture (*colorMap*), with their color varying depending on their distance from the black hole. The intensity of the disk is modulated by parameters like disk height, lighting, and the presence of particles.

3. Skybox

The skybox represents the vast space surrounding the black hole, visualized using a cube map (*galaxy*). The skybox texture is sampled when rays exit the black hole's influence, providing a seamless and visually consistent backdrop**.**

4. Camera and User Interaction:

The camera system allows real-time interaction, enabling users to adjust the viewpoint using mouse controls or predefined angles (*front, top, free view*). It dynamically follows the black hole and uses *lookAt* function to ensure the black hole is always the focal point.

5. Key Shader Parameters:

Several uniform parameters control the scene's visuals and physics, including:

- **Black Hole Mass**: Affects gravitational lensing, and particle movement in the accretion disk.
- **Accretion Disk Parameters**: Customizable settings (e.g., *adiskHeight*, *adiskDensityV*) to adjust the disk's appearance
- **Skybox Textures**: Selection of cube maps to render different 'galaxies'

Challenges and Solutions

Challenge	Solution
Code implementation of complicated gravitational lensing equations	Simplified ray bending equation using Flannelhead's blog[3] to simplify the Schwarzschild metric[9]
Realistic Accretion Disk Rendering	Enhanced accretion disk realism by adding: - 3D noise[1] for texture variation - Dynamic lighting (adiskLit) - Density-based scaling (adiskDensityV, adiskDensityH) - Time-based animation (adiskSpeed) for swirling effect
Performance Optimisation	Improved shader performance by: - Limiting ray marching loop iterations to 30 (experimentation) - Adjusting STEP_SIZE to control ray marching granularity - Adding early exit conditions (e.g., event horizon detection) to terminate loops early when possible

Table 1: Challenges and Solutions in Development

Method Insights

Ray marching was used instead of ray tracing due to the complex physics of black holes, such as ray bending due to gravitational lensing. It works by marching rays through a scene and iteratively advancing them based on the black hole's properties at each step, making it well-suited for visualizing effects like gravitational lensing and accretion disks. Below are the advantages, limitations, and effective ways to use this method in black hole rendering:

1. **Advantages**
    a. **Customizability**: Ray marching offers fine control over light interactions, making it ideal for simulating complex effects like gravitational lensing and the

accretion disk. Adjustable parameters such as ray step size and gravitational acceleration enhance its flexibility.

b. **Real-Time Interaction**: ImGui enables real-time parameter adjustments, allowing users to tweak settings like black hole mass and accretion disk density without recompilation of code

2. **Limitations**
    a. **Computational Expense & Performance Bottlenecks**: Ray marching is resource-intensive. Despite optimizations, maintaining smooth real-time performance can be challenging, especially on lower-end hardware.

3. **Effective Use**
    a. **Post-Processing Effects**: Applying bloom, tone mapping, and dynamic lighting enhances realism, especially for the black hole's bright features. These effects create a cinematic look, blending the scene's elements cohesively.
    b. **Shader Optimization**: Balancing visual fidelity and performance requires optimizing shader code. Limiting ray marching iterations, using early exits at the event horizon, and adjusting step sizes help reduce the computational load.

Learning Resources

A summary of key learning resources related to the project can be found in *Table 2* in *Appendix*. These resources include videos, articles, and code examples that provided valuable insights into concepts like gravitational lensing, black hole simulations, and ray marching techniques.

Conclusion

This project successfully demonstrated the real-time rendering of a black hole, effectively capturing its complex visual phenomena, including gravitational lensing and the accretion disk. By leveraging ray marching, the simulation achieved a visually compelling and interactive experience, with enhanced realism through effects like bloom and tone mapping. This project highlights the power of shaders and custom rendering pipelines in creating scientifically inspired visualizations, pushing the boundaries of real-time graphical realism.

References

[1] Ashima, 3D Noise, GitHub, URL: https://github.com/ashima/webgl-noise/tree/master

[2] CodeProject contributors, Ray Tracing a Blackhole in C#, CodeProject, URL: https://www.codeproject.com/Articles/994466/Ray-Tracing-a-Black-Hole-in-Csharp

[3] Flannelhead, Photons and Blackholes, Mar. 6, 2016, URL: https://flannelhead.github.io/posts/2016-03-06-photons-and-black-holes.html

[4] Interstellar [Movie trailer], Warner Bros., 2014, URL: https://www.youtube.com/watch?v=zSWdZVtXT7E

[5] Narkowicz, Krzysztof, ACES Filmic Tone Mapping Curve, January 6, 2016, URL: https://knarkowicz.wordpress.com/2016/01/06/aces-filmic-tone-mapping-curve

[6] Oseiskar, Ray Paths and Gravitational Lensing, Oseiskar's Black Hole Documentation, URL: https://oseiskar.github.io/black-hole/docs/physics.html

[7] SpaceRip, director. Einstein's Rings and the Fabric of Space, YouTube, Dec. 10, 2013, URL: https://www.youtube.com/watch?v=RI8H4XEs0hw

[8] Walczyk, M, Ray marching. Michael Walczyk, March 31, 2025, URL: https://michaelwalczyk.com/blog-ray-marching.html

[9] Wikipedia contributors, Schwarzschild geodesics, Wikipedia, URL: https://en.wikipedia.org/wiki/Schwarzschild_geodesics

[10] Wikipedia contributors, Spherical coordinate system, Wikipedia, URL: https://en.wikipedia.org/wiki/Spherical_coordinate_system

[11] Wong, Jamie, Ray Marching and Signed Distance Functions, July 15, 2016, URL: https://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions

Appendix

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arccos\frac{z}{\sqrt{x^2+y^2+z^2}} = \arccos\frac{z}{r} = \begin{cases} \arctan\frac{\sqrt{x^2+y^2}}{z} & \text{if } z > 0 \\ \pi + \arctan\frac{\sqrt{x^2+y^2}}{z} & \text{if } z < 0 \\ +\frac{\pi}{2} & \text{if } z = 0 \text{ and } \sqrt{x^2+y^2} \neq 0 \\ \text{undefined} & \text{if } x = y = z = 0 \end{cases}$$

$$\varphi = \operatorname{sgn}(y)\arccos\frac{x}{\sqrt{x^2+y^2}} = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0, \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

Figure 3: Cartesian to Spherical Coordinates Equation

Topic	Description	Format	Reference
Gravitational Lensing	Einstein's Rings and the Fabric of Space: Explains gravitational lensing	Video	[7]
Schwarzschild Geodesics	Describes the motion of test particles in the gravitational field of a central fixed mass.	Wikipedia	[9]
Photons and Blackholes	Explains how light and particles move around a mass, simplified equations used for light bending code	Article	[3]
Ray Paths and Gravitational Lensing	Project example influenced shader code structure	GitHub	[6]
Ray Tracing a Black Hole in C#	Influenced the structure and conceptual approach of shader code.	Article	[2]
Spherical Coordinate System	Obtain cartesian to spherical coordinates function from formula in Figure 3 (Appendix)	Wikipedia	[10]
3D Noise	Apply WebGL 3D Noise to the accretion disk	GitHub	[1]

Ray Marching and Signed Distance Functions	Ray marching and camera movement resource	Article	[11]
Ray Marching Coding Tutorial	A helpful resource for learning the fundamentals of ray marching.	Article	[8]
ACES Filmic Tone Mapping Curve	Code Implementation of Tone Mapping	Article	[5]

Table 2: Learning Resources