I-013

ブラックホール時空における追跡ゲームの改良：円弧テッセレーションによる降着円盤の描画

Improving the chase game in a black hole spacetime: rendering the accretion disk by arc tessellation

山下 義行[1)]
Yoshiyuki Yamashita

1 はじめに

GPU 演算性能の急伸は，ブラックホール時空下のコンピュータグラフィックス（CG）の可能性を大きく拡大させている．加えて新たなシェーダー機能が柔軟なプログラミングをサポートしている．これらの動向を踏まえ，著者は，プレイヤーがブラックホール時空内を宇宙船で自由に移動できるシミュレーションゲームの開発を 10 年前に始めた [1, 2]．現在，以下に述べる課題はあるものも，開発したゲームは PC およびタブレットコンピュータ上で平均 60fps（毎秒 60 枚）以上の描画速度を達成し，ゲームの公開へ向けて大きく前進している．

先行研究 [2] では， object shader/mesh shader [3] を用いた高速テッセレーション（動的プリミティブ分割）を提案した．しかし三角形ポリゴンで定義された降着円盤（ブラックホール周辺の物体が高速に回転しながらブラックホールへ落ちていくときに形成される円盤状のガス体）を描画する場合には，テッセレーション負荷が高く，ゲームとして十分な描画速度には至らなかった．そこで円盤をテッセレーションを必要としないポイントスプライトで代替描画する方法を提案した．これによって十分な描画速度は達成できるものの，あくまでも応急措置であって，このような事情を知らされないゲームプレイヤーに降着円盤の有り様を誤解させる恐れがある．特に小中高生に誤解を与えることは教育上好ましくない．

本研究の課題は，降着円盤を現在，天文学者らによって考えられている描像 [5, 6] に可能な限り近い形で描画し，かつゲームプレイに十分な描画速度を達成することである．その解決法として降着円盤を

- ポイントスプライトよりも表現力が高く，
- ポリゴンよりも高速描画が可能な

円弧（arc）を用いて定義する．光線の湾曲による描画像の歪みは先行研究 [2] の手法を応用し，円弧のテッセレーションで実現する．これによって描画のリアリティと高速描画を両立させる．

本稿の構成は以下の通りである．2 節では先行研究 [2] の描画の問題点を述べる．3 節では円弧のテッセレーションの方法を述べる．4 節では描画例と描画速度を報告する．

2 先行研究による降着円盤の描画

本節では，本研究の背景を概説する．

2.1 ポリゴン描画の問題点

ブラックホール時空における CG の技術課題は，光線がブラックホールの強い重力で曲がり，描画像が歪むことである．ラスタライズ法を相対論的に拡張する先行研究 [2] では，この歪みの描画をポリゴンのテッセレーションで解決する．実際の処理には object shader/mesh shader [3] を用い，処理速度を向上させる．

しかし降着円盤のポリゴン描画（図 1 (1) 参照）ではテッセレーションの計算負荷は大きい．特に円盤が光速に近い速度でブラックホールを周回する場合は，周回しない（またはゆっくりと周回する）場合に比べてポリゴン描画像の歪みが大きく，その分だけテッセレーション処理が増大する．表 1 #1 は，図 1 (1) の CG 画像をデスクトップ PC（Mac Studio 2022）で描画した場合の平均描画速度である．ゲームに必要な描画速度の目安を 60fps とするならば，97 fps は必要速度を上回っているものの，ゲームでは降着円盤以外に敵宇宙船，星野（star field）なども描画せねばならない．

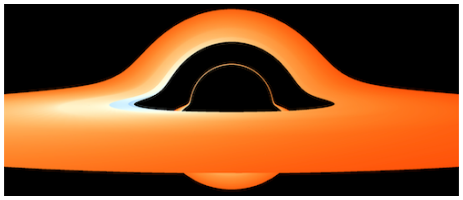
図 2 (1), (2), (3) の最左列は，降着円盤をポリゴンで描画した場合の追跡ゲームの実行画面例である．ゲーム開始から終了までの平均および最悪描画速度は表 2 #1 である．平均描画速度は 66fps であり，円滑なゲームプレイには不足している．実際，動画記録（YouTube [4] No.2 参照）でもぎこちない動きが見られる．

2.2 ポイントスプライト描画の問題点

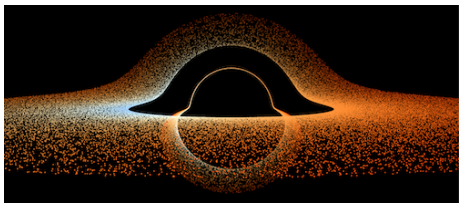
このポリゴン描画の遅さを当面解決するために，先行研究では 10 万個のポイントスプライトで円盤の形状を描画した．図 1 (2) は単体での描画像であり，図 2 の左から 2 番目の列はゲーム実行画面例（YouTube [4] No. 1 参照）である．この場合，単体での描画速度は 497fps（表 1 #2 参照）であり，十分に速い．ゲーム中の平均描画速度は 132fps（表 2 #2 参照）であり，目安の 60fps を大きく超えている．

ポイントスプライトを用いれば速度的にはおおむね満

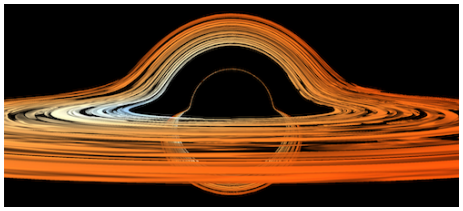
1） 西九州大学デジタル社会共創学環
Digital Society Co-Creation Interdisciplinary, Nishikyushu University.



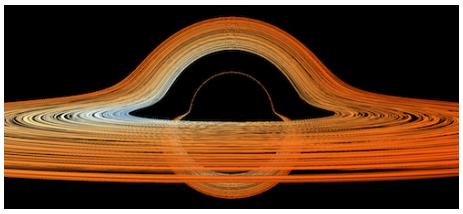
(1) 三角形ポリゴンでの描画



(2) ポイントスプライトでの描画



(3) 円弧 1,100 本での描画



(4) 円弧 4,200 本での描画

図 1　先行研究，本研究における回転する降着円盤の描画例（YouTube [4] No.3，No. 7, No. 8 参照）：(1)，(2) はそれぞれ論文 [2] の図 13，図 12 から転載した．

足できるが，しかし描画像に問題が生じる．ゲームプレイヤーは，円盤を点群で表すことが描画速度を理由とした応急策であるとは知らず，実際の降着円盤が点群で構成されていると誤解する恐れがある．特に小中高校生の誤解は理科教育上問題が多い．ポイントスプライトの数を増やせば遠目には個々の点は目立たなくなるが，円盤に接近した場合には結局，個々の点が大きく描画される（図 2 (3) の左から 2 番目の画像を参照）から，本質的な解決にはならない．

これを解決するためには，描画速度を落とすことなく，世の中に広く知られている降着円盤の像を描画できることが望まれる．

2.3　円弧による描画の提案

降着円盤の CG 画像は今や多くの研究者によって作られている．本研究では，それらの中でも NASA Scientific Visualization Studio 作成の CG 画像 [5] および映画「インターステラー」内の CG 画像 [6] を現在，広く世の中に

表 1　先行研究，本研究における降着円盤単体の平均描画速度：#1，#2 は論文 [2] の表 1 から転載した．#1〜#4 は図 1 (1)〜(4) に対応する．Mac での実行結果である．

#	種別	個数	fps
1	ポリゴン	960	97
2	ポイントスプライト	100,000	497
–	円弧	570	576
3	円弧	1,100	457
–	円弧	2,100	339
4	円弧	4,200	227
–	円弧	8,500	140
–	円弧	17,000	119

表 2　先行研究，本研究におけるゲーム実行時の平均および最悪描画速度：#1，#2，#i1，#i2 は論文 [2] の表 2 から転載した．#1〜#4 の実行の様子は図 2 である．#i1〜#i4 は iPad での実行結果であり，それ以外は Mac での実行結果である．

			fps	
#	種別	個数	平均	最悪
1	ポリゴン	960	66	19
2	ポイントスプライト	100,000	132	25
–	円弧	570	112	24
3	円弧	1,100	109	23
–	円弧	2,100	105	26
4	円弧	4,200	102	25
–	円弧	8,500	100	21
–	円弧	17,000	82	10
i1	ポリゴン	960	53	5
i2	ポイントスプライト	100,000	106	15
i3	円弧	1,100	99	14
i4	円弧	4,200	89	14

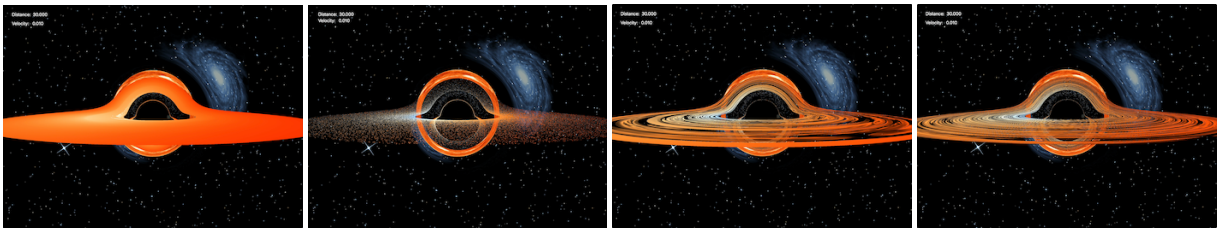
了解されている降着円盤の像の標準モデルとみなす．これらの画像では円盤の周囲に渦上のガスの流れが表現されている．

本研究ではこれをブラックホールの回りに多数の円弧を配置（図 3 参照）して疑似的に表現する方法を提案する．個々の円弧の黒体輻射温度，回転速度は先行研究同様に天文学の知見 [7] に従い計算することとし，円弧の開き角，半径（ブラックホール中心からの距離），幅は乱数を用いて円盤の描画に不自然なアーチファクトが生じないように設定する．

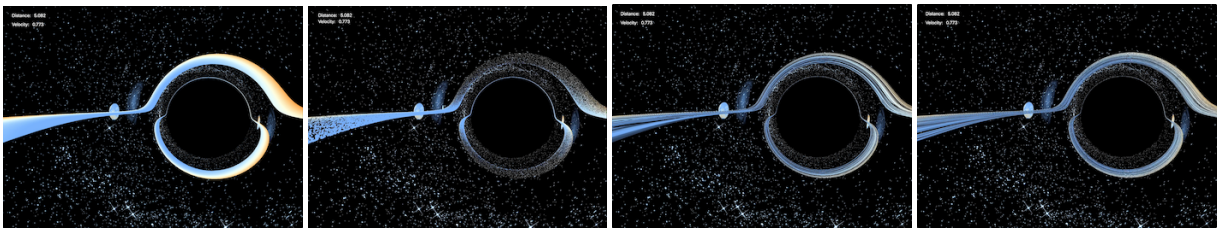
3　円弧のテッセレーション

個々の円弧のテッセレーションは先行研究のポリゴンのテッセレーションと同様に object shader/mesh shader を用いて 2 段階で行う．

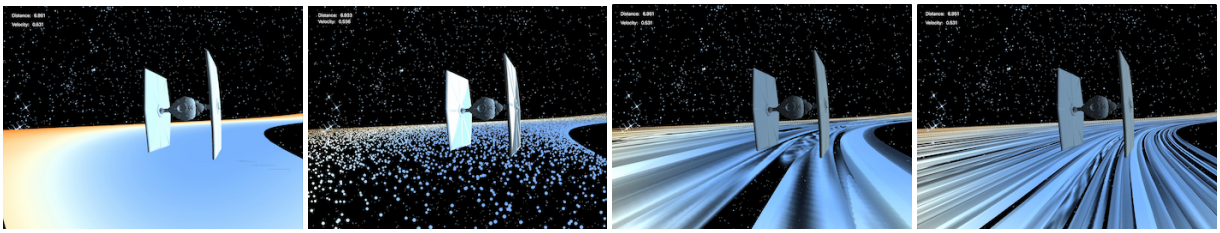
その方法を図 4 を用いて概説する．まず，3 次元空間内のブラックホール近傍に与えられた円弧 P_1P_2 を 2 分割するか否かを考える．そのために円弧の中間点 P_m を求め，さらにこれら 3 点が投影される点 S_1，S_m，S_2 を求める．次に，線分 S_1S_2 の中間点 $\bar{S}_\mathrm{m} = (S_1 + S_2)/2$ と S_m との距離 $\delta = |\bar{S}_\mathrm{m} - S_\mathrm{m}|$ を求め，ある閾値 Δ について



(1) ゲーム開始直後



(2) 加速して降着円盤の内側に突入



(3) 降着円盤の上を背後から宇宙船に接近

図 2　先行研究，本研究における追跡ゲームの実行画面の例（YouTube [4] No.1，No. 2, No. 9，No. 10 参照）：左から順に，降着円盤をポリゴンで描画，ポイントスプライトで描画，円弧 1,100 本で描画，円弧 4,200 本で描画する場合．左から二つは論文 [2] の図 14 から転載した．

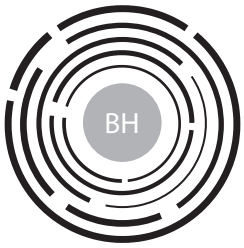


図 3　ブラックホールの回りの円弧：真上から見た場合．個々の円弧の幾何情報には乱数で揺らぎを与える．

次のいずれを適用する．

1. もし誤差 δ が Δ よりも小さいならば，分割は行わず，円弧を線分 S_1S_2 で近似する．
2. さもなくば，円弧 P_1P_2 を円弧 P_1P_m と円弧 $P_\mathrm{m}P_2$ に分割し，それぞれの分割を再起的に繰り返す．

上の再帰処理の停止条件は先行研究と同じとする．object shader と mesh shader の負荷分散も先行研究と同じとする．

テッセレーションが終了したならば，個々の線分に元の円弧の幅を加味して細長い四辺形（2 枚の三角形ポリゴン）を定義し（図 5 参照），ラスタライザへ受け渡す．その際，四辺形の半径方向の辺はブラックホール中心に向かうものとする．フラグメントシェーダでは降着円盤内のガス体の揺らぎをテクスチャで表現する．

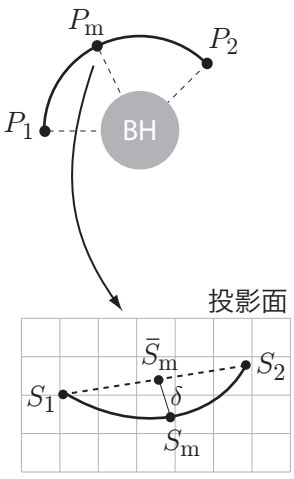


図 4　円弧の分割とテッセレーション近似誤差

4　描画実験

本節では描画像および描画速度を検証する．

4.1　描画像の比較

降着円盤単体の円弧による描画像 2 種類を図 1 (3)，(4) に示す．円弧を設定するに当たり，円弧の本数，幅，円弧間の隙間など様々なパラメータで指定できる．ここでは円弧の数が少ない場合（1,100 本）と多い場合（4,200 本）それぞれにおいて 2.3 節の標準モデルに近い描画像

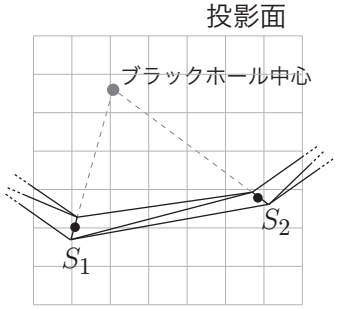


図 5　テッセレーション終了後の三角形ポリゴンによる近似

を得られるパラメータ設定を採用した．なお，円弧の定義に乱数を用いているため，円弧の本数は同一パラメータであっても実行毎に多少変動する．ここで示す本数はおおよその平均本数である．

図 1 (3) の画像では (4) にくらべ，円盤に隙間が大きい．これは円弧の数が少ないためである．隙間を埋めるには円弧の幅を拡げればよいが，単純に拡げると描像が不自然になるから幅には一定の限度があり，結果，隙間が生じる．

ゲームの実行時画面の例を図 2 の右から 2 番目の列，最右の列に示す．プレイヤーが降着円盤から遠いときには円弧数による描画像の違いは目立たないが，プレイヤーが円盤上を滑空する場合には円弧数が少ない場合（図 2 (3) 右から 2 番目の画像参照）にテクスチャの粗が見えてしまう．

4.2　描画速度

表 1 の 3 行目から 8 行目に降着円盤単体の描画速度を示す．同じく表 2 の 3 行目から 8 行目に本研究でのゲーム実行の開始から終了までの平均および最悪描画速度を示す．詳細な分析のため，円弧が 1,100 本，4,200 本以外の場合についても描画速度を測定した．これらはデスクトップ PC（Mac Studio 2022） での実行結果である．

まず，570 本の円弧の場合の単体描画速度はポイントスプライトのそれを超えるが，ゲームの描画速度はそうはならない．複数の被写体を同時に描画する状況下では object shader/mesh shader の動作が vertex shader の動作よりも高コストになるためと推察される．

円弧による単体描画速度は円弧の本数と共に急速に減少する[1)]．しかしゲーム実行では他の被写体の描画処理が合算されるため，単体の描画に比べ，描画速度の減少は緩やかである．たとえば 1,100 本の場合の単体描画速度 457fps は 4,200 本の場合の 227fps へほぼ半減するが，ゲームの描画速度は 109fps から 102fps へと 7%しか減速しない．その意味では円弧の本数を節約して描画速度を稼ぐ戦略は効果的とは言えない．

次に，表 2 の #i1 から #i4 はタブレットコンピュータ（iPad Pro 5th Gen.）でのゲームの描画速度である．PC の場合と同様に円弧での描画速度はポリゴンでの描画速度よりも早く，ポイントスプライトよりも遅い．デスクトップ PC との違いは、タブレットコンピュータでは平均描画速度に比べて最悪描画速度の落ち込みが大きいことである．

これらの結果から，現状，十分に満足できる描画速度とまでは言えないものの，今後のハードウェアの性能向上を期待し，本研究では 4,200 本の場合を当面採用する予定である．

1) 表 1 の関数近似では，円弧による描画速度は円弧の本数の平方根にほぼ反比例している．

5　おわりに

本研究ではポイントスプライトによる描画像がプレイヤーに誤解を与えかねない問題点を円弧の描画で解決する方法を提案した．円弧のテッセレーションはポリゴンのテッセレーションよりも低コストであり，ゲームとして必要最低限の描画性能を達成できた．円弧による降着円盤の像は本研究で想定する標準モデルに近いものになっていると考えている．

今後は描画速度の向上に加え，描画像の表現のさらなる改良を目指す．近い将来に研究成果をゲームとして広く世の中に公開することを計画している．

参考文献

[1] Yamashita, Y.: Implementing a Rasterization Framework for a Black Hole Spacetime, Journal of Information Processing, Vol. 24, No. 4, pp. 690–699 (online), DOI: 10.2197/ipsjjip.24.690 (2016).
[2] 山下義行: ブラックホール時空での追跡ゲームの開発, 情報処理学会論文誌, Vol. 66, No. 7 (2025) 掲載予定.
[3] Apple: Metal Shading Language Specification Version 3.2, Apple (online), available from https://developer.apple.com/metal/Metal-Shading-Language-Specification.pdf (accessed 2024/10/12).
[4] 山下義行: Y’s CG room, YouTube (online), available from https://www.youtube.com/@yscgroom9314 (accessed 2024/10/12).
[5] Francis Reddy : Black Hole Accretion Disk Visualization, NASA Scientific Visualization Studio (2019), available from https://svs.gsfc.nasa.gov/13326 (accessed 2025/5/27).
[6] James, O., Tunzelmann, E. v., Franklin, P. and Thorne, K. S.: Gravitational lensing by spinning black holes in astrophysics, and in the movie Interstellar, Classical and Quantum Gravity, Vol. 32, No. 6, p. 065001 (online), DOI: 10.1088/0264-9381/32/6/065001 (2015).
[7] Kato, S., Fukue, J. and Mineshige, S.: Black-Hole Accretion Disks, Kyoto University Press (1998).