

Projektbericht MoleHole

Gabriel Martin¹ und Leonhard Neuhold¹

¹IT-Abteilung, HTL Krems

Abstract

Dies ist der Projektbericht zum MoleHole Projekt im Rahmen des Faches Projektmanagement der IT-Abteilung an der HTL Krems. Dieses Projekt beschäftigt sich mit der Visualisierung und Simulation von schwarzen Löchern. Dafür soll Rücksicht auf verschiedenste physikalische Effekte wie den Gravitationslinseneffekt oder die Schwarzschildmetrik genommen werden. Weiters werden für die Darstellung dieser Erscheinungen Technologien wie OpenGL und in Spezialisierung Ray Tracing und Ray Marching verwendet.

Inhaltsverzeichnis

1 Einleitung	1
2 Projektdokumentation	2
2.1 Projektentstehung und -planung	2
2.1.1 Projektentstehung	2
2.1.2 Recherche und Planung	3
2.2 Projektumsetzung und Ausblick	5
2.2.1 Innovationsgehalt und Einzigartigkeit	5
2.2.2 Realisierung	5
2.2.3 Ergebnisse	8
2.2.4 Entwicklungspotenzial	12
2.3 Bericht des Projektkoordinators	14
3 Einsatz von KI	15
3.1 Einsatz in der Programmierung	15
3.2 Einsatz in der Dokumentation	15
4 Literaturverzeichnis	16
5 Abbildungsverzeichnis	19

1 Einleitung

Die Physik hinter schwarzen Löchern besteht aus komplizierten Theorien und Beobachtungen, die außergewöhnliche Techniken erfordern [1–3]. Obgleich es schwierig ist, die Mathematik und die Ideen hinter diesen Konzepten zu verstehen, ist es möglich, ein allgemeines Verständnis dieser Objekte aufzubauen. Dafür setzt das MoleHole Projekt auf eine interaktive Oberfläche, die es Benutzern ermöglicht, schwarze Löcher und deren Auswirkungen auf die Umgebung darzustellen und verschiedene Szenarien auszuprobieren. Um diese unterschiedlichen Darstellungen zu erreichen, kann man Effekte wie den *Gravitationslinseneffekt*, die *Raumzeitkrümmung*, die *Akkretionsscheibe*, den *Dopplereffekt* und die *Gravitation* beobachten. Dieses Dokument schildert die wichtigsten Fakten und Erkenntnisse aus dem Projektverlauf. Dabei wird auf die Idee der Simulation schwarzer Löcher, die anfängliche Planung und das Projektmanagement eingegangen. Weiters wird beschrieben, wie dieses Projekt einen

bleibenden Eindruck in der Gesellschaft hinterlassen kann und wie es die Wissenschaftskommunikation verbessern soll. Danach werden einige technische Details erklärt, die im Hintergrund für die Visualisierung und Simulation der schwarzen Löcher verantwortlich sind. Es werden auch einige Grafiken zu sehen sein, die verdeutlichen sollen, wie die Applikation, die das Ergebnis des Projekts bildet, aussieht. Damit kann man erkennen, welche verschiedenen Möglichkeiten die Software bietet. Ein wichtiger Teil ist außerdem der Ausblick und das Potenzial, das in diesem Projekt steckt. Zum Abschluss wird ein Bericht über den Projektverlauf vom Projektkoordinator, Gabriel Martin, verfasst.

2 Projektdokumentation

In diesem Teil des Berichtes findet man die Dokumentation zum MoleHole Projekt. Es sind alle wichtigen Informationen und Erkenntnisse aus dem Projektverlauf zusammengefasst. Es soll ein möglichst klares Bild der entstandenen Applikation vermittelt werden.

2.1 Projektentstehung und -planung

Der folgende Abschnitt soll sich mit der Phase vor dem Beginn des eigentlichen Projektes und der darauffolgenden Planung auseinandersetzen.

2.1.1 Projektentstehung

In dieser Sektion werden die Projektidee und das allgemeine Ziel des Projektes behandelt. Dabei wird auf die äußereren Umstände sowie die eigenen Ideen eingegangen.

Projektidee

Im Unterrichtsgegenstand *Informationstechnische-Projekte-Labor (ITPL)* an der HTL Krems [4] hatte das Projektteam die Aufgabe, ein Projekt nach dessen eigenen Vorstellungen zu planen und durchzuführen. Die beiden Mitglieder des Projektteams waren frustriert mit dem Verlauf deren früherer Projekte und wollten etwas Neuartiges entwickeln. Daraus entsprang die Idee, schwarze Löcher virtuell und visuell darzustellen und in Echtzeit zu simulieren. Im Zuge der Einigung auf ein Gesamtziel versuchte das Projektteam, diese Vorstellung in einzelne Teile zu gliedern. In Abbildung 1 sind die originalen Notizen des Projektteams zu sehen, die bei der Überlegung der Projektziele entstanden sind. Diese Ziele stellen die übergeordneten Gesamtausmaße des Projektes dar, welche das Team erreichen möchte. Im Zuge der besseren Lesbarkeit und genaueren Dokumentation werden diese sogenannten funktionalen Anforderungen nochmals in höherer Genauigkeit aufgelistet:

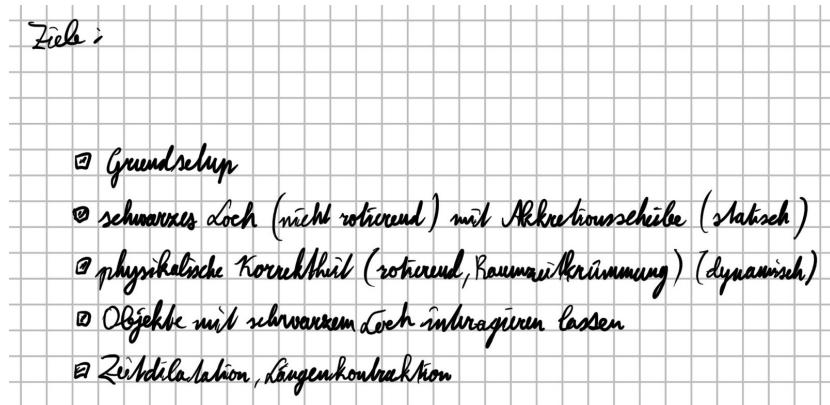


Abbildung 1: Originale Notizen bei der Aufstellung der Projektziele

- **Basis mit Dear ImGui:** Als Basis wird eine *C++ Applikation* aufgesetzt, welche die *Library Dear ImGui* nutzt [5], um dem Endnutzer die Möglichkeit zu bieten, verschiedenste *Parameter einzustellen*, als auch sich im drei dimensionalen Raum zu bewegen.
- **Visualisierung:** Zum besseren Verständnis schwarzer Löcher trägt vor allem die Visualisierung bei. Deshalb sollen der *Ereignishorizont*, die *Photonensphäre*, die Akretionsscheibe und die Raumzeitkrümmung dargestellt werden. Zudem sollen auch noch verschiedene Parameter eines schwarzen Loches angepasst werden können. Das sind die Masse m , Position x^i , Geschwindigkeit v^i sowie der *Spin-Parameter* a . Diese Punkte sollen mithilfe des *Shader-Frameworks OpenGL* erfüllt werden.
- **Simulation:** Um über die reine Darstellung hinaus auch die unvorstellbaren Kräfte schwarzer Löcher zu erkennen, soll es auch die Möglichkeit geben, eine Simulation zu konfigurieren und danach anzusehen. In dieser soll es möglich sein, Phänomene wie die Anziehungskräfte zwischen verschiedenen Objekten zu erfahren. Außerdem wird die Möglichkeit, andere Objekte wie z.B. Sterne, Planeten und Raumschiffe einzufügen und mit dem schwarzen Loch interagieren zu lassen, einen Teil der Implementierung ausmachen.
- **Physikalische Korrektheit:** Damit ein Endnutzer auch lernen kann, wie schwarze Löcher in der realen Welt funktionieren, sollen möglichst genaue Theorien über schwarze Löcher angewendet werden. Zu diesen gehören z.B. die *Kerr-Metrik* zur Berechnung der Raumzeitkrümmung in der Nähe rotierender schwarzer Löcher oder der Dopplereffekt zur Veränderung des ausgesandten Lichts.

Allgemeines Ziel

Weiters bildete sich über diese formulierten funktionalen Anforderungen auch ein allgemeines Ziel, das lautet:

Die komplizierte Physik unseres Kosmos sollte jedem Menschen so einfach und gut verständlich wie möglich präsentiert werden!

Um den Worten des Astronomen Florian Freistetter gerecht zu werden [6], soll den Menschen Wissenschaft beigebracht werden, indem man sie auf eine Reise einlädt, die Komplexität verschiedenster Theorien weglässt und vor allem ein Vertrauen aufbaut. Dieses Projekt soll das umsetzen und damit eine Hilfe in der Kommunikation von Wissenschaft mit der allgemeinen Bevölkerung darstellen.

These

Es war nicht nötig, für dieses Projekt eine These zu formulieren, da es sich nicht um eine herkömmliche wissenschaftliche Arbeit handelt, bei der eine Hypothese formuliert wird und im weiteren Verlauf der Arbeit mithilfe wissenschaftlicher Methoden bestätigt oder wieder verworfen wird. Dieses Projekt soll vielmehr ein Produkt hervorbringen, dass es den Menschen ermöglicht, auf einfache Art und Weise Wissenschaft zu erfahren. In diesem Projekt werden bei Erfüllung der Projektziele große Teile der Errungenschaften der Astronomie und Kosmologie der letzten 150 Jahre umgesetzt und versucht, dem Endnutzer möglichst verständlich zu präsentieren.

2.1.2 Recherche und Planung

In dieser Sektion wird die Recherche und Planung des Projektes beschrieben. Es wird dafür auf die verschiedenen Quellen eingegangen sowie die Art der Planung erklärt.

Informationsgewinn

Die Quellen für die Recherche und den Informationsgewinn sind so zahlreich, dass sie nicht in einer Liste aufgeführt werden können, ohne den Rahmen dieses Dokumentes zu sprengen oder die Lesbarkeit

zu beeinträchtigen. Deshalb werden hier nur einige Beispiele angeführt. Als wichtige Ressourcen zählen einige Projekte, die als Inspiration für die Arbeit dienen. Vor allem die Programmierung dieser Projekte war von großer Relevanz, da sie bei der Implementierung verschiedener Effekte helfen. Die Ressourcen sind:

- **Black Hole Raytracer - Ross Ning:** Dieses Projekt hat anfänglich als hauptsächliche Quelle für die Programmierung der Visualisierung schwarzer Löcher gedient. Es ist eine gute Darstellung, auch wenn die Physik nicht perfekt umgesetzt ist [7].
- **Black Hole Raytracer - Hydrogendeuteride:** In diesem Projekt sind vor allem der Dopplereffekt und die *Schwarzkörperstrahlung* genau umgesetzt worden [8].
- **Black Hole Raytracer - Thomas Kamminga:** Dieses Projekt ist vor allem aufgrund der korrekten physikalischen Implementierung eine gute Hilfe [9].
- **Black Hole Raytracer - Eliot Han:** Die Dokumentation dieser Ressource war eine Hilfestellung, indem auf einer verlinkten Website die Lösung für die Implementierung des Gravitationslinseneffektes gut erklärt ist [10, 11].
- **Andere:** Außerdem gibt es auch noch einige andere Projekte oder Beiträge, die von der Visualisierung schwarzer Löcher handeln, sie sind in [12–17] aufgezählt.

Viele dieser Projekte können auf GitHub betrachtet werden, während auch auf Reddit etwas zu finden war. Auch Bücher waren eine ordentliche Hilfestellung bei der Implementierung. Diese sind:

- **Spacetime and Geometry: An Introduction to General Relativity:** Dieses Buch ist eine umfangreiche Einführung in die *Allgemeine Relativitätstheorie (ART)* und erklärt die dafür notwendigen mathematischen Grundlagen [1].
- **Schwarze Löcher: Der Schlüssel zum Verständnis des Universums:** In diesem Buch werden in populärwissenschaftlicher Weise bestimmte Phänomene und Theorien rund um schwarze Löcher erklärt [18].
- **Andere:** Es gibt natürlich auch noch viele andere Bücher zu dieser Thematik, allerdings wurden viele nicht für das Projekt verwendet oder es wurde nur wenig Wissen entnommen, wie aus [19, 20].

Insgesamt ist die Recherche für ein Projekt dieser Komplexität äußerst wichtig und dementsprechend viele verschiedene Quellen wurden hierfür herangezogen. Neben den oben genannten wichtigen Ressourcen wurden auch einige wissenschaftliche Artikel, Dissertationen, Videos und Webinhalte genutzt, um an das notwendige Wissen zu gelangen.

Planung des Projektablaufes

Die Planung des Projektes erfolgt agil, was nicht bedeutet, dass auf eine Planung im Vorfeld verzichtet wurde. Vielmehr wurde sich am Anfang die Zeit genommen, um die groben Ziele aufzustellen – siehe 2.1.1 – und Rahmenbedingungen zu schaffen sowie das Aufsetzen eines *GitHub-Repositories* und der *Projektmanagementsoftware YouTrack* [21, 22]. Darüber hinaus werden kleinere Aufgaben für jede Iteration eines sogenannten *Sprints* [23] eingeteilt. Die untergeordneten Projektziele werden im Laufe des Projektes angepasst und hinzugefügt oder gelöscht. Es wird eine Mischung der agilen Projektplanungsframeworks *Scrum* [24] und *Kanban* [25] – *Scrumban* [26] – ausgeführt. Dies wird eingehalten, indem Rollen für die Mitglieder des Projektteams festgelegt wurden, wöchentliche Meetings abgehalten werden, alle drei Wochen ein *Product Increment* erfolgt und alles über ein *Kanban-Board* verwaltet wird. Zudem werden mittels einer *Knowledge-Base* alle Meetings dokumentiert und andere wichtige Informationen festgehalten. Diese Art, ein Projekt zu planen und zu führen, ermöglicht eine flexible Arbeitsweise und macht besonders im Falle dieses Projektes Sinn. Dies kommt daher, dass die beiden Richtungen der Programmierung und der Physik miteinander kombiniert werden müssen und dies einen ständigen Paradigmenwechsel sowie eine Überarbeitung alter Konzepte erfordert. Da alle

Beteiligten des Projektes großes Engagement zeigen, wird auch außerhalb des Unterrichts gearbeitet und versucht, die aktuellen Ziele zu erfüllen.

2.2 Projektumsetzung und Ausblick

Dieser Abschnitt beschreibt die wesentlichen Merkmale dieses Projektes und soll die Realisierung und die aktuellen Ergebnisse sowie das Entwicklungspotenzial erläutern.

2.2.1 Innovationsgehalt und Einzigartigkeit

In dieser Sektion wird erklärt, warum dieses Projekt einzigartig ist und einen eigenen innovativen Charakter enthält.

Innovationscharakter

Dieses Projekt kann als innovativ oder außergewöhnlich angesehen werden, da schwarze Löcher für einen Großteil der Bevölkerung ein sehr unbekanntes Thema sind und eher einschüchternd wirken, wenn man die komplexe Mathematik dahinter betrachtet. Außerdem geht es um besonders abstrakte Konzepte, da bis heute erst zwei echte Bilder eines schwarzen Loches angefertigt werden konnten [2, 3]. Demnach ist das Projekt innovativ, weil es sich mit dieser komplizierten Thematik auseinandersetzt. Darüber hinaus wird auch versucht, anderen Menschen ein Verständnis für das Thema zu vermitteln, indem eine interaktive Applikation entwickelt wird, die es ermöglicht, spielerisch und experimentell die komplexen Vorgänge in der Nähe eines schwarzen Loches zu erleben.

Rechtslage

Das Projekt ist ein OpenSource-Projekt und der gesamte Quellcode ist auf GitHub ersichtlich [21]. Zudem wurde es unter der GNU General Public License v3.0 (GPL-3.0) veröffentlicht [27].

Einzigartigkeit

Es gibt ohnehin wenig frei zugängliche Projekte, die als Ziel die Visualisierung schwarzer Löcher verfolgen. Jedoch ist bei diesen Projekten nie eine Benutzerinteraktion vorgesehen und schränkt somit die Bedienung dieser Applikationen stark ein. Genau in dieser Lücke findet sich das MoleHole Projekt wieder, indem es die Echtzeitsimulation schwarzer Löcher mit einer interaktiven Benutzeroberfläche kombiniert.

2.2.2 Realisierung

Diese Sektion befasst sich mit der Implementierung der aktuellen Version der Applikation und soll diese Schritt für Schritt beschreiben.

Technologiestack

Der Großteil der Applikation wurde in C++ entwickelt. Die einzige Programmiersprache, die zusätzlich verwendet wurde, ist GLSL für die Shader-Programmierung. Die Programmiersprachen sind also sehr einfach gehalten. Im Gegensatz dazu werden einige Bibliotheken verwendet, um nicht alles selbst entwickeln zu müssen. Alle verwendeten Bibliotheken sind:

- **OpenGL Libraries:** GLAD, GLFW und GLM für Parameter in GLSL und andere Utilities [28–30]
- **Dear ImGui:** Standardisierte Library für das *User Interface* [5]

- **Spdlog:** Library für das Logging [31]
- **PhysX:** Physiksimulationen verschiedenster Art [32]
- **Stb_image:** Verarbeiten von Bildern [33]
- **ImGuizmo:** Utility-Library für die Selektion von Objekten [34]
- **Tinygltf:** Für das *File-Processing* von .gltf-Dateien [35]
- **ImGui-Node-Editor:** Framework für einen *Node Editor* [36]
- **Yaml-cpp:** Verarbeitung von .yaml-Dateien [37]
- **Nativefiledialog-extended:** *File-Dialogs* für den Export von Videos und Bildern [38]

Zusätzlich zu diesen Technologien, gibt es auch einige Tools und Programme, die für die Entwicklung verwendet wurden. Dabei wurden CLion und YouTrack von JetBrains [22, 39] und Git [40] als *Versionskontrolle* eingesetzt.

Rendering Techniken

Die wichtigsten Rendering Techniken, die in der Applikation verwendet werden, sind Ray Tracing und Ray Marching. Beim Ray Tracing geht von der Kamera aus bis zu einem Schnittpunkt mit einem Objekt ein Lichtstrahl aus und wird dort reflektiert und zur Lichtquelle zurückverfolgt. Dies wird für eine große Anzahl an Schnittpunkten durchgeführt, bis schlussendlich das ganze Objekt sichtbar ist. Dies wird dann am Bildschirm dargestellt. Beim Ray Marching nutzt man auch Lichtstrahlen, die von der Kamera ausgehen, aber man geht hier Schritt für Schritt und rechnet nicht sofort den Schnittpunkt mit einem Objekt aus. Grundsätzlich ist diese Methode also rechenintensiver, aber sie ermöglicht es, gekrümmte Lichtstrahlen abzubilden. Diese gekrümmten Lichtstrahlen sind dafür verantwortlich, dass in der Nähe eines schwarzen Loches komplexe visuelle Effekte auftreten. Die Lichtstrahlen werden durch die Gravitation abgelenkt und dieser sogenannte Gravitationslinseneffekt wird durch das Ray Marching in der Applikation simuliert. Da die Rechenleistung eines normalen Computers für das Ray Marching nicht ausreicht, muss man einen Kompromiss eingehen. Man verzichtet auf die exakte Darstellung der Objekte und beschäftigt sich nur mit der Simulation von Lichtstrahlen, die sehr nahe an einer großen Masse vorbeigehen. Dadurch wird gewährleistet, dass die gewünschten optischen Effekte zu sehen sind, jedoch auch nicht zu viel Genauigkeit verloren geht, da die Lichtstrahlen in größerer Entfernung nicht mehr wirklich sichtbar abgelenkt werden. Für die Software werden also beide Rendering Techniken eingesetzt: Ray Tracing um Lichtstrahlen bis zu einer gewissen Entfernung zu schwarzen Löchern zu simulieren und Ray Marching um die Lichtstrahlen in der Nähe von schwarzen Löchern zu berechnen.

Physikalische Effekte

Die physikalischen Effekte, die bis zum jetzigen Stand in der Software realisiert worden sind, bilden die Grundlage für die Simulation schwarzer Löcher.

Der wichtigste Effekt betrifft den Gravitationslinseneffekt, der sich durch die starke Raumzeitkrümmung in der Nähe eines schwarzen Loches ergibt. Hierbei werden Lichtstrahlen in der Nähe einer großen Masse stärker abgelenkt als in größerer Entfernung. Dies entspricht dann einem ähnlichen Effekt wie bei einer herkömmlichen *Konvexlinse*. Um diesen Effekt visuell darzustellen, ist es nötig die *Nullgeodätische* auszurechnen, welche den Weg des Lichts um ein schwarzes Loch angibt [1]. Man nehme an, dass $x(\lambda)$ den Weg eines Objektes durch Raum und Zeit und Γ das *Christoffel-Symbol* angibt. Damit ergibt sich nach Lösung von

$$\frac{d^2x^\rho}{d\lambda^2} + \Gamma_{\mu\nu}^\rho \frac{dx^\mu}{d\lambda} \frac{dx^\nu}{d\lambda} = 0 \quad (1)$$

genau dieser Weg [1, 19]. Für diese Lösung gibt es in der *Schwarzschild-Metrik* eine vereinfachte Form, die in der Applikation tatsächlich verwendet wird. Die Kraft $\vec{F}(r)$ ergibt sich aus

$$\vec{F}(r) = \frac{3h^2\hat{r}}{2r^5} \quad (2)$$

mit einer Konstante h , dem Radius r und der Basis \hat{r} [11, 17].

Die Theorie, die vorerst zur Simulation der Gravitation verwendet wird, ist die *Gravitationstheorie* von Isaac Newton. Diese Theorie ist schon sehr alt, funktioniert aber immer gut für die Berechnung der Anziehungskräfte zwischen Objekten mit geringen Massen. G sei die Gravitationskonstante, M und m die Massen zweier Objekte und \vec{r} der Abstand zwischen den Zentren der beiden Objekte. Die Gravitationskraft \vec{F}_G berechnet sich dann durch [41]

$$\vec{F}_G = \frac{GMm}{\vec{r}^2}. \quad (3)$$

Für die Visualisierung der Raumzeitkrümmung wird die Schwarzschild-Metrik verwendet. Diese wird zur Berechnung der Krümmung in der Nähe einer Punktmasse verwendet. Es seien die Distanz zwischen zwei Ereignissen ds , der Schwarzschildradius R_S , die Entfernung vom Zentrum der Masse R sowie die Dimensionen der Raumzeit dt , dr und $d\Omega$ gegeben. Dann beschreibt

$$ds^2 = -\left(1 - \frac{R_S}{R}\right)dt^2 + \frac{1}{\left(1 - \frac{R_S}{R}\right)}dr^2 + r^2d\Omega^2 \quad (4)$$

die Raumzeitkrümmung in der Nähe einer Punktmasse [1, 18, 19].

Ein weiterer wichtiger Effekt ist die richtige Temperaturskalierung in der Akkretionsscheibe und die damit verbundene Lichtemission, die vom menschlichen Auge oder einem Bildsensor als Farbe erkannt wird. Man nehme an, dass G die Gravitationskonstante, σ die Stefan-Boltzmann-Konstante, M die Masse des schwarzen Loches, \dot{M} die Akkretionsrate, R_i der innere Radius der Akkretionsscheibe und R der aktuelle Radius in der Akkretionsscheibe ist. Die Temperaturverteilung $T(R)$ in der Akkretionsscheibe lässt sich dann durch

$$T(R) = \left[\frac{3GM\dot{M}}{8\pi\sigma R^3} \left(1 - \sqrt{\frac{R_i}{R}}\right) \right]^{\frac{1}{4}} \quad (5)$$

beschreiben [42].

Forschungsmethoden

Die verwendete Methodik, um Erkenntnisse zu gewinnen, ist die Empirie. Sie dient der Verbesserung der Applikation. Es wurde ein Problem identifiziert und danach an einer vermeintlichen Lösung gearbeitet. Nachdem man eine Lösung auf dem Papier entwickelt hatte, wurde diese in der Applikation implementiert und ausprobiert. Durch die Überprüfung der neuen Implementierung konnte eine Entscheidung getroffen werden, ob die Lösung zielführend war oder nicht. Genau so musste das Projektteam die Performance verbessern. Es wurde festgestellt, dass die Software die gewünschte Leistung nicht lieferte, also musste sich etwas überlegen werden. Deswegen wurde mithilfe einiger Zeichnungen an der Tafel eine mögliche Lösung erarbeitet. Diese bezieht sich auf die oben genannte Kombination aus Ray Tracing und Ray Marching. Nach der Implementierung dieser Lösung konnte festgestellt werden, dass die Performance signifikant verbessert wurde.

Ein weiteres Mittel für die Gewinnung wichtiger Erkenntnisse ist die Arbeitsweise mit Sprints wie in 2.1.2 erklärt. Sie ermöglicht die Reflexion über die vergangene Arbeit und eine kontinuierliche Verbesserung des Projektes. Außerdem sind in diesem Format regelmäßige Erweiterungen der Software geplant - sogenannte Product Increments - die vom agilen Projektmanagement vorgegeben sind und andauernd für eine verbesserte Version des Produktes sorgen.

Weiters war die Recherche ein wichtiges Mittel, um an nötige Informationen zu gelangen und die aktuellen Ergebnisse und Implementierungen zu überprüfen. Hierfür wurden wissenschaftliche Artikel, Bücher und Videos wie zum Beispiel [1, 18, 43] verwendet.

2.2.3 Ergebnisse

Diese Sektion beschäftigt sich mit dem Fortschritt, der bis heute erreicht werden konnte. Dabei werden einige Bilder zu sehen sein und anhand einiger Zahlen der Umfang beschrieben.

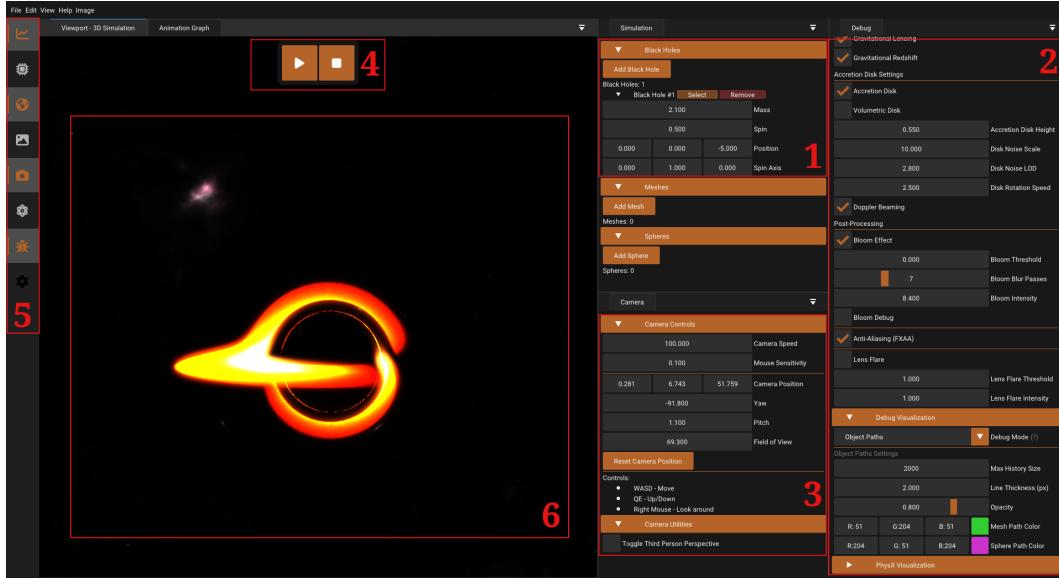


Abbildung 2: Beispiel des User-Interfaces der Applikation.

Ergebnisse in visueller Darstellung

Die folgende Liste wird die Bereiche des User-Interfaces (UIs) der Applikation erklären und minimal auf die Funktionsweise der Unterpunkte eingehen. In Abbildung 2 ist ein Beispiel des UIs abgebildet. Die nummerierten Bereiche werden nachstehend erläutert:

- 1 - Objekte:** In diesem Bereich des Menüs ist es möglich, Objekte hinzuzufügen, zu löschen und zu bearbeiten. Es können drei verschiedene Arten an Objekten verwendet werden: schwarze Löcher, Sterne & Planeten sowie verschiedene 3D-Modelle aus .gltf-Dateien. In dieser Szene wurde beispielhaft ein schwarzes Loch hinzugefügt. Es ist zu erkennen, dass für ein schwarzes Loch die Parameter Masse m , Spin a , Position x^i sowie die Drehachse a^i anpassbar sind.
- 2 - Visualisierungen:** Dieses Fenster ermöglicht die Anpassung verschiedenster Effekte, die im Prozess des Renderings verwendet werden. Dazu zählen die Aktivierung des Gravitationslinseneffektes, der gravitativen Rotverschiebung, des Volumetric Cloud Renderings sowie des Dopplereffektes. Weiters lassen sich die Höhe der Akkretionsscheibe, die Skalierung der *Noise-Funktion*, die für die Darstellung der Akkretionsscheibe genutzt wird, die Details in der Scheibe, als auch die Geschwindigkeit ω mit der sich die Scheibe dreht, einstellen. Weiters gibt es die Möglichkeit, den *Bloom-Effekt* zu aktivieren und einige Einstellungen dafür, die man verändern kann. Zusätzlich ist es auch möglich, das *Anti-Aliasing* einzuschalten, welches für eine verbesserte Qualität des Bildes der Szene sorgt. Die Einstellungen, die in diesem Menü aber die größten Möglichkeiten bieten, sind die verschiedenen *Debug-Layer*, die angezeigt werden können. Es gibt hier die Auswahl zwischen den *Object Paths*, welche die Pfade der Objekte während einer Simulation anzeigen und dem *Gravity Grid*, welches die Raumzeitkrümmung in einem dreidimensionalen Gitternetz darstellt. Diese Debug-Layer sind in Abbildung 5 zu sehen.
- 3 - Kamera:** Der Bereich für die Konfiguration der Kamera ist hier zu finden. Die meisten Einstellungen in diesem Menüpunkt werden meistens nicht gebraucht, da sich auch mit den Tasten W, A, S, D, Q und E fortbewegt werden kann. Allerdings können Fortbewegungsgeschwindigkeit

und das Sichtfeld angepasst werden. Zudem sind hier auch die Einstellungen für die Perspektiven zu finden. Wenn die *Third-Person-Perspektive* aktiviert wird, ist es möglich ein 3D-Modell als Kamera-Objekt auszuwählen und dieses durch eine Szene zu bewegen. In Abbildung 4b kann dies angesehen werden.

- **4 - Simulation:** Diese beiden Icons bieten die Möglichkeit, eine Simulation zu starten, zu pausieren oder zu stoppen. Wenn die Simulation gestoppt wird, werden auch alle Objekte in den Ursprungszustand zurückgesetzt.
- **5 - Seitenleiste:** Die Seitenleiste soll dem Benutzer eine bessere Übersicht geben und die Funktion bereitstellen, die Menüs ein- und ausblenden zu können. Weiters sind hier auch die Einstellungen untergebracht, die für das Setzen eines standardmäßigen Export-Pfades und das Anpassen der Schriftgröße und Schriftart eine Möglichkeit bieten.
- **6 - Szene:** Dies ist der Bereich, in dem die hauptsächliche Simulation und Visualisierung eingebettet ist und somit das Kernstück der Software ist. Hier kann man alle Einstellungen als finales Bild ansehen und eine Simulation beobachten.



(a) Rendering eines schwarzen Loches mit weißer Akkretionsscheibe (b) Rendering eines schwarzen Loches mit Akkretionsscheibe und realistischer Lichtemission.

Abbildung 3: Rendering schwarzer Löcher mit Akkretionsscheibe und Gravitationslinseneffekt.

Die beiden Rendering-Ergebnisse in Abbildung 3 zeigen jeweils ein einziges schwarzes Loch. Bei dem linken Bild in Abbildung 3a ist die Akkretionsscheibe als weiß zu erkennen. Das kommt daher, da dieses Rendering in einem früheren Stadium des Projektes aufgenommen worden ist und die Temperaturskalierung nicht den realen Verhältnissen entsprochen hat. Im rechten Bild in Abbildung 3b kann man eine genauere Verteilung der Temperatur in der Akkretionsscheibe erkennen, welche sich in Abhängigkeit der Entfernung zum Zentrum des schwarzen Loches ändert. Diese Temperaturverteilung wird in der Gleichung 5 mathematisch beschrieben.

Für die nachfolgenden Bilder wurde eine Grafik aus dem Internet verwendet, um die restlichen Objekte besser hervorzuheben. Dieses Bild ist ein Foto der Milchstraße von Jcomp [44].

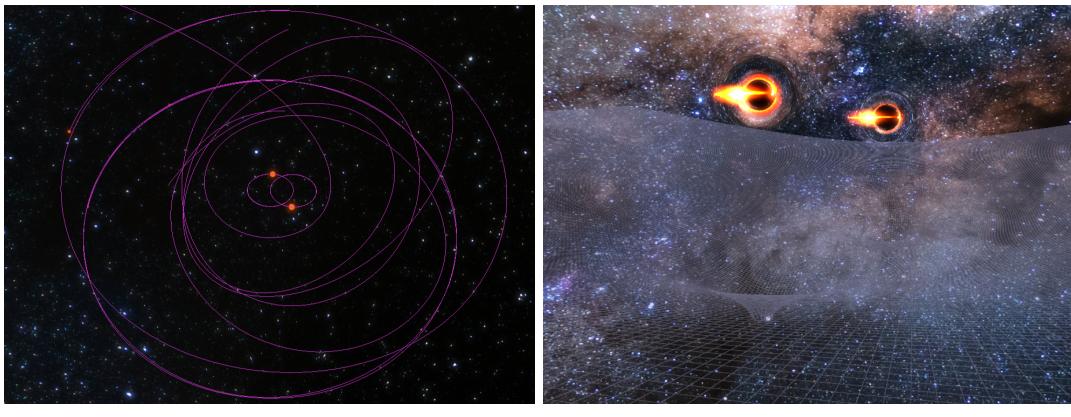
In der Software wurden zwei verschiedene Kameraperspektiven umgesetzt: die First-Person-Perspektive und die Third-Person-Perspektive. In Abbildung 4 kann man die beiden im Vergleich betrachten. Für die Third-Person-Perspektive kann man ein Objekt auswählen, das man dann steuern kann. Dieses Objekt wird mit einem zusätzlichen Würfel, welcher eine Art Crosshair darstellen soll, in der Applikation gerendert.



(a) First-Person-Perspektive

(b) Third-Person-Perspektive

Abbildung 4: Vergleich der First-Person- und Third-Person-Perspektive.



(a) Object Paths Debug-Layer

(b) Gravity Grid Debug-Layer

Abbildung 5: Beispiele der Debug-Layer zur Visualisierung komplexer Phänomene.

Weiters ist auch das Anregen der Vorstellungskraft ein wichtiger Punkt. Hierfür gibt es die Debug-Layer, welche in Abbildung 5 zu sehen sind. Diese Visualisierungen sollen dem Benutzer helfen, sich die physikalischen Effekte besser vorstellen zu können. Da es leider nicht möglich ist, sich eine 4-dimensionale Raumzeit vorzustellen, muss dem Gehirn ein wenig auf die Sprünge geholfen werden. Diese Tricks nutzt die Applikation, um ein besseres Verständnis für die komplexen und abstrakten Konzepte der Physik zu vermitteln.

Ergebnisse in Zahlen

Eine weitere gute Methode, um den Fortschritt eines Projektes zu messen, ist die Angabe bestimmter Kennzahlen, um eine gewisse Größe und Komplexität zu beschreiben. Die wichtigsten Zahlen betreffen die Performance der Applikation bei verschiedenster Auslastung durch das Rendering einer verschiedenen Anzahl an Objekten.

Für den Test der Performance wurde ein Acer Predator Triton 500 Laptop mit einem Intel Core i7-10750H und einer Nvidia GeForce RTX 2080 Super verwendet [45]. Die Ergebnisse sind am 25.01.2026 aufgenommen worden und sind in Abbildung 6 zu sehen. Es wurden für jedes Szenario insgesamt 10 Stichproben genommen und danach der Durchschnittswert berechnet. Aus den Daten lässt sich klar erkennen, dass vor allem schwarze Löcher einen ordentlichen Rechenaufwand benötigen. Für 3D Modelle (Meshes) und Sterne & Planeten (Spheres) ist die Performance nahe am Wert ohne Objekte.

Insgesamt hat das Projekt bis zum 25.01.2026 197 Commits auf GitHub zustande gebracht. Dabei wurden 1404 Zeilen Code in der Sprache GLSL und 12966 Zeilen in der Sprache C++ geschrieben. Das Projekt wurde bis jetzt von einem Team aus zwei Personen im Durchlauf von 5 Sprints realisiert.

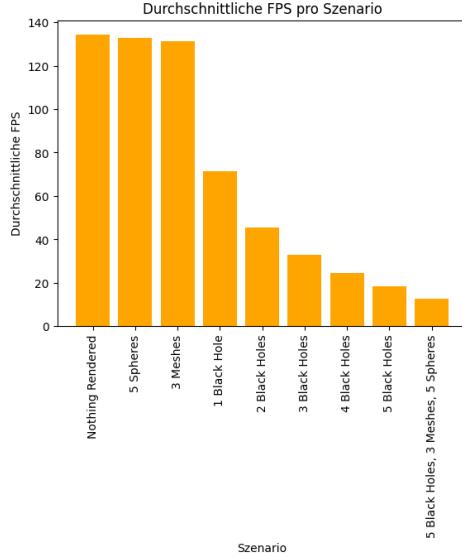


Abbildung 6: Ergebnisse des Tests der Performance in verschiedenen Szenarien.

Prüfen der abgeschlossenen Aufgaben

Das Projektteam setzt sich alle Ziele am Anfang eines sogenannten Sprints [23] und schreibt diese in der Projektmanagementsoftware YouTrack [22] nieder. Dort werden allgemein die Ziele formuliert, mit einer Priorität versehen, der Arbeitsaufwand anhand von *Story Points* geschätzt und ein Verantwortlicher festgelegt. Um zu überprüfen, ob das Projektteam die Ziele erfüllt, werden diese in der Applikation von den Mitgliedern des Projektteams ausprobiert. Da dies keine besonders zuverlässige Methode zur Bestimmung der erfüllten Ziele ist, gibt es am Ende eines jeden Sprints ein *Sprint-Review Meeting*, bei dem das Team ihre Ergebnisse dem Projektbetreuer sowie einem ausgewählten Kunden (ein Mitschüler) vorstellt und den Sprint somit abschließt [23]. Danach werden erneut Ziele für die nächste Iteration überlegt.

Erreichte Ziele

Im Vergleich zu 1 und 2.1.1 sind bis jetzt einige Ziele erreicht worden. Das Grundsetup der Applikation wurde erfolgreich durchgeführt und die wichtigsten Basisfunktionen wie das Laden und Speichern von Szenen sowie das Verschieben der Fenster sind funktionstüchtig. Es ist dem Benutzer auch möglich, das UI zu bedienen und sich im dreidimensionalen Raum fortzubewegen. Die Visualisierung mit OpenGL wurde auch schon erfolgreich implementiert. Hierbei kann ein statisches schwarzes Loch mit einer Akkretionsscheibe beobachtet werden. Außerdem kann der Gravitationslinseneffekt, die gravitative Rotverschiebung, der Dopplereffekt und auch die Photonensphäre gesehen werden. Diese Effekte sind vor allem in der Abbildung 3b gut zu erkennen. Für die Simulation hat der Benutzer die Möglichkeit, andere Objekte wie Kugeln und 3D-Modelle hinzuzufügen und miteinander interagieren zu lassen. Es ist hierfür möglich, die Gravitation zu beobachten, die zwischen allen verschiedenen Objekten wirkt. Weiters können die Objekte miteinander kollidieren und für den Fall, dass ein schwarzes Loch mit einem Objekt zusammenstößt, wird dieses aus der Szene entfernt.

2.2.4 Entwicklungspotenzial

Es steckt noch einiges an Potenzial in diesem Projekt und das Team plant eine regelmäßige Erweiterung der Software, um diesem gerecht zu werden. Demnach werden in dieser Sektion alle potenziellen Erweiterungen des Projektes aufgezählt.

Gesellschaftliche Auswirkungen

Wie bereits in 2.1 erwähnt ist das Gesamtziel des Projektes eine verbesserte Wissenschaftskommunikation durch eine Applikation mit der man experimentell und spielerisch komplexe Vorgänge unseres Universums erleben und erlernen kann. Das Produkt ist so konzipiert, dass es die Möglichkeit gibt, eigene Simulationen zu erstellen und in diesen zu experimentieren, aber auch über Tooltips und Dokumentation Genaueres über die simulierten physikalischen Effekte zu erfahren. Weiters ist die Applikation als OpenSource-Software frei auf GitHub verfügbar und wird dies in Zukunft auch bleiben, was den gesellschaftlichen Nutzen erheblich verbessert, weil es jeder, der einen PC besitzt, ausprobieren kann. Insgesamt lässt sich feststellen, dass es dem Projektteam ein Anliegen ist, mit dieser Applikation die Wissenschaftskommunikation zu verbessern und damit mehr Menschen den Zugang zur Wissenschaft zu erleichtern.

Kooperationen

Aktuell gibt es bei diesem Projekt keine bestehenden Kooperationen, jedoch würde das Projektteam in Zukunft welche anstreben. Vor allem in der Zukunft möchte das Projektteam die Applikation weiter verbessern, indem im zukünftigen Studium wichtige Erkenntnisse gesammelt werden und mit professionellen Wissenschaftlern zusammengearbeitet wird. Zusätzlich wird mithilfe von Kooperationen auch die Entwicklung weiterer Forschungsarbeiten angestrebt.

Releases

Wenn die Entwicklung des Projektes weiterhin große Fortschritte erzielt, ist eine Veröffentlichung des Projektes zum Beispiel auf Steam anzuvisieren. Aktuell kann das Projekt zwar schon benutzt werden, da es ein öffentliches GitHub Repository mit dem Code gibt, aber es ist ein wenig technisches Wissen notwendig, um die Applikation zu verwenden. Wenn diese auf einer Vertriebsplattform wie Steam veröffentlicht werden kann, würde das den Zugang zu der Software erleichtern. Insgesamt kann die Aussage getroffen werden, dass das Projekt in Zukunft offiziell veröffentlicht werden soll, aber noch viel Arbeit notwendig ist, um dieses Ziel zu erreichen. Hierfür muss vor allem noch eine umfangreiche Dokumentation angefertigt werden und die technischen Potenziale sollten möglichst gut ausgeschöpft sein. Hinzu kommt die Entfernung aller Fehler und Unannehmlichkeiten aus der Software.

Technische Potenziale

Das mit Abstand wichtigste physikalische Phänomen, das in der Applikation noch umgesetzt werden muss, ist die Kerr-Metrik, welche im Jahre 1963 vom neuseeländischen Physiker Roy Kerr als Lösung der Feldgleichungen aus der ART aufgestellt worden ist [1]. Hierfür wird allerdings Wissen aus der ART benötigt, das sich das Projektteam noch nicht angeeignet hat. Deshalb wird es noch einige Zeit dauern, bis dieses Feature verwirklicht werden kann. Nichtsdestotrotz beschäftigt sich das Team mit diesem Effekt, da dieser extrem relevant für eine akkurate Darstellung eines schwarzen Loches ist. Mit der Kerr-Metrik folgen einige andere wichtige Phänomene aus der ART, die als mögliche Ziele des Projektes im Hinterkopf behalten werden sollen, jedoch erst nach Umsetzung der Kerr-Metrik implementiert werden können. Diese Effekte sind das Frame-Dragging, die Zeitedilatation, die Längenkontraktion sowie die Gravitation im Allgemeinen. Vor allem Zeitedilatation und Längenkontraktion sind wichtig für eine genaue Simulation schwarzer Löcher, da für Beobachter außerhalb der unmittelbaren Umgebung eines schwarzen Loches Objekte nahe dem Ereignishorizont sich immer langsamer bewegen und letztendlich verblassen [19]. Hierbei muss natürlich auch die Gravitation implementiert werden, die der wichtigste Bestandteil der zukünftigen Ziele sein wird. Es soll dabei die aktuell verwendete Simulation der Gravitation nach der Gravitationstheorie von Isaac Newton [41] durch die Gravitationstheorie von Albert Einstein, die ART, ersetzt werden. Mit dieser Theorie wäre es möglich, Objekte, die sich sehr nahe an großen Massen befinden, genauestens zu simulieren. Generell lässt sich sagen, dass die Theorie der Gravitation von Isaac Newton in der Nähe von schwarzen Löchern nicht funktionieren wird, da die

Raumzeit hier besonders stark gekrümmmt ist [1, 19]. Aus der Gravitationstheorie lässt sich ein weiterer Effekt ableiten, der für eine genaue Simulation von Objekten in der Nähe großer Massen von Relevanz ist: die Gezeitenkraft [46]. Mit der Implementierung dieser Kraft wäre es möglich, Himmelskörper wie zum Beispiel Millers Planet aus dem Film Interstellar [20, 47] physikalisch möglichst genau zu simulieren. Aus technischer Perspektive soll die visuelle Darstellung der Objekte auch noch verbessert werden. Hierfür wird das *Volumetric Cloud Rendering* verwendet werden, um die Akkretionsscheibe eines schwarzen Loches detailreicher darzustellen. Diese Rendering-Technik kümmert sich um die Beleuchtung wolkenartiger Gebilde, da Licht in Wolken gestreut und abgelenkt wird, sodass nicht genau das Licht den Beobachter erreicht, das ursprünglich in seine Richtung ausgesandt worden ist [43, 48]. Außerdem wird eine Partikel Simulation angestrebt, sodass man sehen kann, wie sich über die Zeit eine Akkretionsscheibe um ein schwarzes Loch bildet. Abgesehen von der technischen Simulation, ist es auch das Ziel, Wissenschaft zu kommunizieren. Daher ist auch geplant, Erklärungen und Beschreibungen mittels Tooltips und detaillierte Berichte in einem Menü unterzubringen. Diese Erklärungen sollen auch nicht auf Mathematik verzichten, diese aber verständlich erklären und wenn nötig auf externe Ressourcen verweisen. Somit wäre garantiert, dass interessierte Menschen relativ einfach an die nötigen Ressourcen gelangen können, die notwendig sind, um komplexe physikalische Konzepte zu verstehen.

Wirtschaftliches Potenzial

Der *Entrepreneurship-Charakter* besteht aus der Möglichkeit, durch das Projekt Erfahrungen und Erkenntnisse zu sammeln und diese in zukünftigen Projekten anzuwenden. Der wirtschaftliche Nutzen dieses Projektes besteht also indirekt, indem dieses Projekt als Chance angesehen wird, um Wissen zu erlangen und dieses in zukünftigen Projekten wirtschaftlich erfolgreich anzuwenden. Deshalb wird auch genauestens auf ein sorgfältiges Projektmanagement geachtet, um diese Erfahrung später professionell nutzen zu können. Es ist dem Projektteam wichtig, Wissenschaft für alle kostenlos zugänglich zu machen, weshalb dieses Projekt rein dem Erkenntnisgewinn und der Wissenschaftskommunikation dient.

2.3 Bericht des Projektkoordinators

Dieser Abschnitt bietet einen Überblick des Projektes aus der Sicht des Projektkoordinators. Es wird auf die Teamarbeit sowie die schulische Betreuung eingegangen.

Zusammenarbeit im Team

Die Zusammenarbeit im Team verlief bis jetzt ausgesprochen gut. Beide Mitglieder brachten ihre individuellen Stärken ein und ergänzten sich dadurch ausgezeichnet. Es wurden viele konzeptionelle Herausforderungen gemeinsam besprochen und gelöst. Auch die Zusammenarbeit mit dem Projektbetreuer war stets konstruktiv und hilfreich. Er gibt dem Entwicklungsteam regelmäßig Feedback, welches zur Verbesserung des Projektes beiträgt.

Kompetenzen im Team

Gabriel hat bereits Erfahrung in der Entwicklung von Applikationen im Bereich Computergrafik mit C++ und OpenGL. Dies war besonders hilfreich zu Beginn des Projektes, da er das Grundgerüst der Applikation schnell aufsetzen konnte. Er ist auch in den Bereichen Software Architektur und Performanceoptimierung sehr kompetent.

Leonhard bringt viel Wissen im Bereich der theoretischen Physik mit, welches für die korrekte Umsetzung der physikalischen Effekte essenziell ist. Zudem ist auch er ein talentierter Programmierer mit der Fähigkeit, sich schnell in neue Technologien einzuarbeiten.

Beide Mitglieder verfügen über viel Allgemeinwissen im Bereich moderner Softwareentwicklungsmethoden, was die Organisation und Planung des Projektes erleichtert hat.

Viele der Grundkenntnisse der Physik und Informatik wurden im Rahmen der Ausbildung an der HTL Krems erworben. Vertiefende Kenntnisse in den Bereichen Computergrafik, Softwareentwicklung und theoretische Physik wurden durch eigenständige Recherche und das Studium von Fachliteratur erlangt.

Aufgabenverteilung

Die Aufgabenverteilung im Team erfolgte größtenteils nach den individuellen Stärken und Interessen der Mitglieder. Gabriel übernahm hauptsächlich die Implementierung des Applikationsgerüsts, die Integration von OpenGL und die Optimierung der Rendering-Performance. Leonhard konzentrierte sich auf die Implementierung der physikalischen Modelle, die Berechnung der Raumzeitkrümmung und die Simulation der Interaktionen zwischen Objekten. Beide Mitglieder arbeiten jedoch eng zusammen und unterstützten sich gegenseitig bei der Lösung von Problemen.

In vielen Bereichen wie UI Design, Rendering und Research arbeiten beide Mitglieder gemeinsam, um die bestmöglichen Ergebnisse zu erzielen.

Schulische Projektbetreuung

Der Projektbetreuer, Jürgen Katzenschlager, hatte zwar keine tiefgehenden Kenntnisse im Bereich der Computergrafik oder theoretischen Physik, konnte aber durch seine Erfahrung in der Projektleitung und Softwareentwicklung wertvolle Ratschläge geben. Er half dem Team, den Fokus auf die wichtigsten Ziele zu legen, und unterstützte bei der Planung und Organisation des Projektes. Seine regelmäßigen Reviews und Feedback-Sessions trugen dazu bei, dass das Projekt auf Kurs blieb und die Qualität der Arbeit hoch war.

Koordination

Die Umsetzung des Projektes erfolgt auf der Basis agiler Methoden mit 3 Wochen Sprints, wie in 2.1.2 erläutert. Wöchentliche Status-Meetings werden abgehalten, um den Fortschritt zu besprechen und eventuelle Probleme zu lösen. Am Ende jedes Sprints findet ein Sprint-Review statt, bei dem die erreichten Ziele präsentiert und bewertet werden. Die Aufgaben werden in einem Kanban-Board in YouTrack [22] verwaltet, wo auch der Fortschritt und die Prioritäten der einzelnen Aufgaben festgehalten werden.

Die Entwicklung verlief bisher größtenteils planmäßig, obwohl es natürlich auch einige Herausforderungen gab, besonders im Rendering und Realtime Performance Bereich, welche zu leichten Verzögerungen führten. An wichtigen Entscheidungspunkten fand sich das Team in einem Brainstorming-Meeting zusammen, um die bestmögliche Lösung zu erarbeiten, welche anschließend erfolgreich umgesetzt wurde, wie in 2.2.2 beschrieben.

Die offene Kommunikation, das gegenseitige Verständnis und die gegenseitige Unterstützung haben dazu beigetragen, dass das Team harmonisch zusammenarbeitet.

Zeitaufwand und Kosten

Zum Stand des 22.01.2026 betrug der gesamte Zeitaufwand für das Projekt etwa 175 Stunden. Dies umfasst sowohl die Entwicklungszeit als auch die Zeit für Recherche, Planung und Meetings. Die Kosten für das Projekt waren minimal, da alle benötigten Werkzeuge und Ressourcen kostenlos verfügbar waren.

3 Einsatz von KI

Dieser Abschnitt soll einen Überblick darüber geben, wie dieses Projekt mit der Technologie der künstlichen Intelligenz (KI) umgegangen ist und inwiefern diese eine Rolle im Projektverlauf gespielt hat.

3.1 Einsatz in der Programmierung

In diesem Projekt wird KI zur Steigerung der Effizienz und zur Erlernung verschiedener Techniken genutzt. Dabei werden hauptsächlich die Claude Sonnet Modelle von Anthropic verwendet, die sich als äußerst nützlich für die Entwicklung erwiesen haben [49–51]. Allgemein hat sich der Einsatz von KI-Tools gelohnt und das Projekt vorangetrieben. Vor allem beim Erlernen der Shaderprogrammierung war die KI hilfreich, da sie Code geschrieben hat, den man ausprobieren und lernen konnte. Jedoch haben sich auch Grenzen beim Verwenden dieser Tools gezeigt. Diese bestehen vor allem in vielen Fehlern, die von der KI gemacht werden und danach händisch ausgebessert werden müssen. Auch die Codequalität erfüllt nicht immer die Standards, die sie erfüllen sollte. Aufgrund dieser Erfahrungen mit der neuen Technologie, wird viel Wert auf das Erlernen der Funktionsweise bestimmter Rendering Techniken und physikalischer Prozesse gelegt, damit die Qualität des Codes den Erwartungen des Projektteams gerecht werden kann.

3.2 Einsatz in der Dokumentation

Im Bereich der Projektdokumentation bietet sich ein anderes Bild. Hier wurde bewusst auf KI verzichtet, da die Qualität des geschriebenen Inhaltes die höchste Priorität hatte. Außerdem ist der Einsatz von KI-Tools auch bezüglich des Urheberrechts eine gewisse Grauzone. Deshalb könnten möglicherweise Probleme auftreten, mit denen das Team nichts zu tun haben möchte.

4 Literaturverzeichnis

- [1] Sean M. Carroll. *Spacetime and Geometry: An Introduction to General Relativity*. 1st. Addison-Wesley, 2004.
- [2] Event Horizon Telescope Collaboration et al. “First Sagittarius A* Event Horizon Telescope Results. III. Imaging of the Galactic Center Supermassive Black Hole”. In: *The Astrophysical Journal Letters* 930.2 (May 2022), p. L14. DOI: 10.3847/2041-8213/ac6429. URL: <https://doi.org/10.3847/2041-8213/ac6429>.
- [3] Kazunori Akiyama et al. “First M87 Event Horizon Telescope Results. IV. Imaging the Central Supermassive Black Hole”. In: *The Astrophysical Journal Letters* 875.1 (Apr. 2019), p. L4. ISSN: 2041-8213. DOI: 10.3847/2041-8213/ab0e85. URL: <http://dx.doi.org/10.3847/2041-8213/ab0e85>.
- [4] HTL Krems. URL: <https://www.htlkrems.ac.at/>.
- [5] Ocornut. *Dear ImGui: Bloat-free Graphical User interface for C++ with minimal dependencies*. URL: <https://github.com/ocornut/imgui>.
- [6] Florian Freistetter. *Warum Podcasts ein effektives Medium für die Wissenschaftskommunikation sind*. 2022. URL: <https://www.youtube.com/watch?v=nYsGHXy2AkM>.
- [7] Ross Ning. *A blackhole simulation using OpenGL / C++*. 2020. URL: <https://github.com/rossning92/Blackhole>.
- [8] Hydrogendeuteride. *blackhole ray marching*. 2025. URL: <https://github.com/hydrogendeuteride/BlackHoleRayTracer>.
- [9] Thomas Kamminga. *C++ project that generates images from space-time curved by gravity*. 2022. URL: <https://github.com/thomaskamminga/RayTracingUsingRelativity>.
- [10] Michael Tu Eliot Han Alejandro García Salas. *Raytracer for a Schwarzschild black hole*. 2019. URL: <https://github.com/eliot1019/Black-Hole-Raytracer?tab=readme-ov-file>.
- [11] Michael Tu Eliot Han Alejandro García Salas. *Black-Hole-Raytracer*. URL: <https://eliot1019.github.io/Black-Hole-Raytracer/>.
- [12] James. *geodesic_raycasting*. 2024. URL: https://github.com/20k/geodesic_raytracing.
- [13] Liam Clegg. *Black hole universal simulation interface engine (a real-time black hole ray tracer)*. 2024. URL: <https://github.com/cleggacus/bhusie>.
- [14] Riccardo Antonelli et al. *Starless is a CPU black hole raytracer in numpy suitable for both informative diagrams and decent wallpaper material*. 2015. URL: <https://github.com/rantonels/starless>.
- [15] Burgao. *My first Kerr black hole simulation with C++*. URL: https://www.reddit.com/r/Physics/comments/113c97c/my_first_kerr_black_hole_simulation_with_c/.
- [16] sirxemic. *Interactive simulation of a wormhole and a black hole similar to those shown in the movie Interstellar*. 2019. URL: <https://github.com/sirxemic/Interstellar>.
- [17] Riccardo Antonelli et al. *How to draw a Black Hole*. URL: <https://rantonels.github.io/starless/>.
- [18] Jeff Forshaw Brian Cox. *Schwarze Löcher: Der Schlüssel zum Verständnis des Universums*. 1st. Kosmos, 2024.
- [19] Charles W. Misner, Kip S. Thorne, and John Archibald Wheeler. *Gravitation*. Reprinted by Princeton University Press, 2017, ISBN 978-0-691-17779-3. San Francisco: W. H. Freeman and Company, 1973. ISBN: 978-0-7167-0344-0.
- [20] Kip S. Thorne. *The Science of Interstellar*. The Science of Interstellar. Foreword by Christopher Nolan. W. W. Norton Company, 2014. ISBN: 9780393351378.

- [21] Leonhard Neuhold Gabriel Martin. *MoleHole*. 2026. URL: <https://github.com/g-martin772/MoleHole/tree/master>.
- [22] JetBrains. *YouTrack - Leistungsstarkes Projektmanagement für alle Ihre Teams*. URL: <https://www.jetbrains.com/de-de/youtrack/>.
- [23] Scrum. *What is a Sprint?* URL: <https://www.scrum.org/resources/what-is-a-sprint-in-scrum>.
- [24] Wikipedia. *Scrum — Wikipedia, die freie Enzyklopädie*. [Online; Stand 31. Dezember 2025]. 2025. URL: <https://de.wikipedia.org/w/index.php?title=Scrum&oldid=262320775>.
- [25] Wikipedia. *Kanban (Entwicklung) — Wikipedia, die freie Enzyklopädie*. [Online; Stand 31. Dezember 2025]. 2025. URL: [https://de.wikipedia.org/w/index.php?title=Kanban_\(Entwicklung\)&oldid=262396453](https://de.wikipedia.org/w/index.php?title=Kanban_(Entwicklung)&oldid=262396453).
- [26] Wikipedia. *Scruban — Wikipedia, die freie Enzyklopädie*. [Online; Stand 31. Dezember 2025]. 2024. URL: <https://de.wikipedia.org/w/index.php?title=Scruban&oldid=250873747>.
- [27] Free Software Foundation. *GNU General Public License*. URL: <https://www.gnu.org/licenses/gpl-3.0.html>.
- [28] David Herberth. *Multi-Language Vulkan/GL/GLES/EGL/GLX/WGL Loader-Generator based on the official specs*. URL: <https://github.com/Davidde/glad>.
- [29] GLFW. *A multi-platform library for OpenGL, OpenGL ES, Vulkan, window and input*. URL: <https://github.com/glfw glfw>.
- [30] icaven. *OpenGL Mathematics (GLM)*. URL: <https://github.com/icaven/glm>.
- [31] Gabi Melman. *Fast C++ logging library*. URL: <https://github.com/gabime spdlog>.
- [32] NVIDIA-Omniverse. *NVIDIA PhysX SDK*. URL: <https://github.com/NVIDIA-Omniverse/PhysX>.
- [33] Sean Barrett. *stb single-file public domain libraries for C/C++*. URL: <https://github.com/nothings/stb/tree/f1c79c02822848a9bed4315b12c8c8f3761e1296>.
- [34] Cedric Guillemet. *Immediate mode 3D gizmo for scene editing and other controls based on Dear ImGui*. URL: <https://github.com/CedricGuillemet/ImGui>.
- [35] Syoyo Fujita. *Header only C++11 tiny glTF 2.0 library*. URL: <https://github.com/syoyo/tinygltf/tree/81bd50c1062fdb956e878efa2a9234b2b9ec91ec>.
- [36] Michał Cichoń. *Node Editor built using Dear ImGui*. URL: <https://github.com/thedmd/imgui-node-editor>.
- [37] Jesse Beder. *A YAML parser and emitter in C++*. URL: <https://github.com/jbeder/yaml-cpp>.
- [38] Bernard Teo. *Cross platform (Windows, Mac, Linux) native file dialog library with C and C++ bindings, based on mlabbe/nativefiledialog*. URL: <https://github.com/btzy/nativefiledialog-extended>.
- [39] JetBrains. *CLion - Plattformunabhängige IDE für C und C++*. URL: <https://www.jetbrains.com/de-de/clion/>.
- [40] Software Freedom Conservancy. *Git*. URL: <https://git-scm.com/>.
- [41] Isaac Newton. *Philosophiae Naturalis Principia Mathematica*. Londini: Jussu Societatis Regiae ac Typis Josephi Streater, 1687.
- [42] H.C. SPRUIT. “Accretion Disks”. In: (). URL: <https://wwwmpa.mpa-garching.mpg.de/~henk/pub/disksn.pdf>.
- [43] Sebastian Lague. URL: <https://www.youtube.com/watch?v=4Q0cCGI6x0U&t=7s>.

- [44] jcomp. *Beautiful galactic core of milky way with rho ophiuchi cloud complex. Long exposure photograph*. URL: https://www.freepik.com/free-photo/beautiful-galactic-core-milky-way-with-rho-ophiuchi-cloud-complex-long-exposure-photograph_7809954.htm#fromView=keyword&page=1&position=0&uuid=620e97f2-daa4-4543-aef1-d77d53605bfb&query=Hdri+milky+way.
- [45] Acer. *Predator Triton 500 PT515-52-742D Gaming-Notebook*. URL: <https://www.acer.com/at-de/predator/laptops/triton/500/pdp/NH.Q6WEV.001>.
- [46] Xiaotian Zhang and Sijie Gao. *Geodesic completeness, curvature singularities and infinite tidal forces*. 2025. arXiv: 2507.04616 [gr-qc]. URL: <https://arxiv.org/abs/2507.04616>.
- [47] Christopher Nolan. *Interstellar*. 2014. URL: <https://www.imdb.com/title/tt0816692/>.
- [48] Shruti Singh and Shantanu Kumar. *Machine Learning-Driven Volumetric Cloud Rendering: Procedural Shader Optimization and Dynamic Lighting in Unreal Engine for Realistic Atmospheric Simulation*. 2025. arXiv: 2502.08107 [cs.GR]. URL: <https://arxiv.org/abs/2502.08107>.
- [49] Anthropic. “System Card: Claude Sonnet 4.5”. In: *Anthropic* (2025). URL: <https://www.anthropic.com/clause-sonnet-4-5-system-card>.
- [50] Anthropic. “System Card: Claude Opus 4 Claude Sonnet 4”. In: *Anthropic* (2025). URL: <https://www.anthropic.com/clause-4-system-card>.
- [51] Xidan Song et al. “Beyond Accuracy: A Geometric Stability Analysis of Large Language Models in Chess Evaluation”. In: (2025). arXiv: 2512.15033 [cs.AI]. URL: <https://arxiv.org/abs/2512.15033>.

5 Abbildungsverzeichnis

1	Originale Notizen bei der Aufstellung der Projektziele	2
2	Beispiel des User-Interfaces der Applikation.	8
3	Rendering schwarzer Löcher mit Akkretionsscheibe und Gravitationslinseneffekt.	9
4	Vergleich der First-Person- und Third-Person-Perspektive.	10
5	Beispiele der Debug-Layer zur Visualisierung komplexer Phänomene.	10
6	Ergebnisse des Tests der Performance in verschiedenen Szenarien.	11