



FIGURE 1

Software-Projekt Raytracer

Florian Bertscher, Christian Kasper, Lukas Jeckle,
Leon Musliu, Dennis Welsch

Team: RayForge

Winter Semester 2024/25

Technische Hochschule Ulm
Software-Projekt
Prof. Dr. R. Lunde

Table of Contents

1. Kontextanalyse	3
1.1 Einleitung.....	3
1.2 Motivation	3
1.3 Vision	3
1.4 Projektkontext.....	4
2. Anwendungsdomäne	5
3. Domänenmodell.....	24
4. Anforderungsdefinition.....	27
4.1 Funktionale Anforderungen	28
4.2 Nicht funktionale Anforderungen	31
5. Anwendungsfälle.....	33
6. Abläufe	33
7. Benutzerschnittstellen	34
7.1 3D – Szene JSON	34
8. Graphische Gestaltung und Nutzungskonzept	37
8.1 Hauptbildschirm.....	37
8.2 Einstellungen (gestrichen).....	42
9. Sources	46
9.1 Contents	46

1. Kontextanalyse

1.1 Einleitung

In der modernen Computergrafik sind realistische Licht- und Schatteneffekte sowie Reflexionen von zentraler Bedeutung. Ziel dieses Softwareprojekts ist die Entwicklung eines leistungsfähigen Raytracers, der benutzerdefinierte 3D-Szenen verarbeitet und nahezu fotorealistische Bilder erzeugt. Durch die systematische Durchführung des Projekts können die Teammitglieder praktische Erfahrungen in allen Phasen der Softwareentwicklung sammeln. Dabei stehen die Vertiefung der Kenntnisse in Anforderungsmanagement, Design, Implementierung und Test im Vordergrund. Das Projekt fördert zudem Teamarbeit und Kommunikationsfähigkeiten in einem Team von 6-8 Personen. Dieses Lastenheft legt die Anforderungen fest, um die technischen und organisatorischen Ziele erfolgreich zu erreichen.

1.2 Motivation

Nach wochenlanger Planung und strategischer Überlegung ist es unser Ziel, in den Jahren 2024/25 einen leistungsfähigen Raytracer zu entwickeln. Dieses Projekt wird es uns ermöglichen, als Team zusammenzuarbeiten und dabei die individuellen Stärken jedes Mitglieds zu nutzen. Durch den Austausch von Wissen und Erfahrungen werden wir nicht nur unsere Fähigkeiten erweitern, sondern auch den Zusammenhalt im Team stärken. Unser gemeinsames Ziel ist es, einen Raytracer zu schaffen, der die Marktstandards übertrifft und einen echten Mehrwert für unsere Nutzer bietet. Der Fokus liegt dabei weniger auf dem individuellen Erfolg, sondern auf dem kollektiven Ergebnis und dem Spaß, den wir gemeinsam bei der Entwicklung haben.

1.3 Vision

Wie soll das fertige System ungefähr aussehen?

Das fertige System wird ein Raytracer sein, der in der Lage ist, nahezu fotorealistische Darstellungen von Dreiecksszenen zu erzeugen.

➔ Diffuses Raytracing

Welche Features soll es haben?

Die geplante Software für den Raytracer wird eine Vielzahl von leistungsfähigen Features umfassen. Zunächst wird sie die Möglichkeit bieten, Oberflächenmodelle einzulesen, die aus Dreiecken zusammengesetzt sind, und zwar im weit verbreiteten .obj/.mtl-Format, was eine einfache Integration von komplexen 3D-Objekten ermöglicht. Anwender können die Perspektive auf das modellierte Objekt frei wählen und verschiedene Ansichten und Kamerawinkel erkunden. Zudem haben sie die Freiheit, punktuelle Lichtquellen beliebig in der Szene zu platzieren, um realistische Beleuchtungseffekte und Schatten zu erzeugen. Eine integrierte 3D-Vorschau wird es den Benutzern ermöglichen, die Ansichten und Beleuchtungen in Echtzeit zu definieren und zu visualisieren, bevor die endgültige Bildberechnung erfolgt. Die Bildberechnung selbst erfolgt durch die Anwendung eines Raytracing-Algorithmus, der das Lichtverhalten in der Szene realistisch simuliert, einschließlich Reflexionen,

Brechungen und Schatten. Schließlich werden die gerenderten Bilder sowohl auf dem Monitor angezeigt als auch in gängigen Grafikformaten wie PNG oder JPEG gespeichert, sodass sie für die weitere Nutzung oder Bearbeitung bereitstehen. Diese Kombination aus Funktionen wird eine leistungsfähige und benutzerfreundliche Softwarelösung bieten, die den Anforderungen von Entwicklern und Grafikern gerecht wird.

In welchen Szenarien soll es eingesetzt werden?

Unternehmen können den Raytracer nutzen, um Produkte in einer realistischen Umgebung darzustellen, was bei Marketingmaterialien und Online-Shops hilfreich ist, um potenziellen Käufern ein besseres Verständnis der Produkte zu vermitteln.

1.4 Projektkontext

Wie passt das Projekt in die durchführende Organisation?

Ziel der Organisationsform für dieses Projekt ist es, zu erkennen und zu entscheiden, welche verschiedenen Rollen und Fachbereiche innerhalb eines Projektes existieren und zu planen, wie diese am effektivsten zusammenarbeiten können. Dabei treten uns folgende Fragen auf wie: Welches Mitglied aus dem Team kann welche Stärken und Kenntnisse in unser Projekt einbringen und welche Rollen/Charaktere können in unserem Projekt verteilt werden.

Welche Stakeholder sind daran beteiligt?

Man unterscheidet zwischen interne und externe Stakeholder. Hier bei unserem Raytracer sind die internen Stakeholder der Auftragsgeber (Eigentümer) und Mitarbeiter beteiligt. Die externen Stakeholder sind hierbei unsere Endnutzer, die unseren Raytracer nutzen.

Welche möglichen Weiterentwicklungen oder Folgeprojekte sind denkbar?

Eine sinnvolle Weiterentwicklung des Raytracers wäre die Integration von Physically Based Rendering (PBR), um Materialien und Lichtquellen physikalisch korrekter darzustellen und so die Realitätsnähe der gerenderten Szenen zu erhöhen. Zusätzlich könnte der Einsatz von Künstlicher Intelligenz dazu dienen, Renderprozesse zu optimieren und automatisch Licht- und Schatteneffekte anzupassen, was die Benutzerfreundlichkeit verbessert und Renderzeiten verkürzt.

2. Anwendungsdomäne

Vorlage:

BEGRIFF	Camera_Component
BESCHREIBUNG	Beschreibt eine Camera, mit ihren unterliegenden Attributen.
IST-EIN	Komponente
KANN-SEIN	Szenenbestandteil
ASPEKT	Die Attribute „fov“, „aspect_ratio“, „near_clip“ und „far_clip“ repräsentieren grundlegende Parameter einer Kamera.
BEMERKUNG	Der Begriff Kamera beschreibt ein Objekt, dass das Sichtfeld des Users auf das Objekt darstellt.
BEISPIELE	<pre> "cameraentity": [{ "name": "Camera", "uuid": "163fe421-7c52-41f4-a2b8-7cb77f88566f", "Translation": { "position": { "x": 10, "y": 250, "z": 400 }, "rotation": { "x": -35, "y": 0, "z": 0 } }, "components": { "CameraComponent": { "fov": 60, "aspectRatio": 1.77, "nearClip": 0.1, "farClip": 10000 } } }] </pre>

BEGRIFF	Render_Component
BESCHREIBUNG	Ein Renderobjekt das alle nötigen Attribute besitzt zum Darstellen.
IST-EIN	Zentrale Komponente der Szene, und stellt das anzuzeigende Objekt mit grundlegenden Attributen dar
KANN-SEIN	Szenenbestandteil
ASPEKT	Besitzt die eindeutige UUID object_UUID, um das Objekt zu identifizieren, sowie material_UUID, um die zugehörigen Materialien aufzulösen.
BEMERKUNG	Der Begriff beschreibt ein Objekt, dass man Rendern möchte.
BEISPIELE	<pre> "renderentity": [{ "name": "Chess-Teapot", "uuid": "63c45451-2a3b-4454-a54d-0242c2d68ee3", "Translation": { "position": { "x": 0, "y": 0, "z": 0 }, "rotation": { "x": 0, "y": 0, "z": 0 }, "scale": { "x": 1, "y": 1, "z": 1 } }, "components": { "RenderComponent": { "objUUID": "4213aa10-c3f3-4505-a257-ea52d89beb6f", "matUUID": "455cb61c-7974-4484-ac7e-85a7a0a58622" } } }] </pre>

BEGRIFF	Light_Component
BESCHREIBUNG	Eine Lichtquelle, die alle wichtigen Attribute hält.
IST-EIN	Zentrale Komponente der Szene, und stellt die Lichtquelle mit grundlegenden Attributen dar.
KANN-SEIN	Szenenbestandteil
ASPEKT	Die Attribute „intensity“ und „color“ in „r g b“ beschreiben die Eigenschaften einer Lichtquelle, wobei intensity die Helligkeit und color die Farbe des Lichts angibt.
BEMERKUNG	Der Begriff beschreibt ein Objekt, dass zur Beleuchtung in der Szene benötigt, wird beim Raytracing.
BEISPIELE	<pre> "lightentity": [{ "name": "Light-1", "uuid": "b7d03d91-52ad-49b2-8c48-cd6a7e9ae822", "Translation": { "position": { "x": -50, "y": 250, "z": 0 } }, "components": { "LightComponent": { "intensity": 0.5, "color": { "r": 255, "g": 0, "b": 0 } } } }] </pre>

BEGRIFF	Converter
BESCHREIBUNG	Ist eine Übersetzungsstruktur, um unsere Matrizen an der Eigen Umgebung anzupassen.
IST-EIN	Übersetzungsstruktur
KANN-SEIN	Für den Benutzer nicht sichtbar.
ASPEKT	Bearbeiteten die Ressourcen und Komponente unserer Hauptszene.
BEMERKUNG	Der Begriff bezeichnet eine Übersetzungsstruktur um unsere Matrizen von GLM auf Eigen
BEISPIELE	Für den Benutzer nicht sichtbar.
BEGRIFF	Base_Component
BESCHREIBUNG	Ist die Überklasse der Unterliegenden Komponenten und verwaltet die gemeinsamen Attribute.
IST-EIN	Oberklasse, „Vatterklasse“
KANN-SEIN	Zentrale Speicherverwaltung der unterliegenden Komponenten
ASPEKT	Die Attribute umfassen die eindeutige UUID (uuid), den Komponententyp (type), den Namen (name) sowie optional die Position, Rotation und Skalierung.
BEMERKUNG	Der Begriff bezeichnet die Oberklasse, der Komponente Camera-, Light- und Render-Komponenten.
BEISPIELE	Beispiele der Umsetzung sieht man an den äußersten Attributen in unserem .JSON-Files.
BEGRIFF	Material_Resource
BESCHREIBUNG	Eine Material Resource repräsentiert die Eigenschaften und Zuordnungen eines Materials in einer Rendering Umgebung. Sie enthält Informationen wie Farbeigenschaften, Beleuchtungsparameter, Transparenz sowie zugehörige Texturen.
IST-EIN	Die Material_Resource ist eine spezialisierte Klasse, die von Base_Resource abgeleitet ist und spezifische Funktionen für Materialeigenschaften bereitstellt.
KANN-SEIN	Ein Material mit bestimmten Farben (ambient, diffuse, specular, etc.) Eine Ressource, die auf Texturen verweist. Eine Datenstruktur für Beleuchtungs- und

	Shader-Parameter (z. B. Transparenz, Glanz)
ASPEKT	<p>Die Material_Resource organisiert Materialdefinitionen und ihre zugehörigen Parameter. Dies schließt:</p> <p>Materialeigenschaften: Farben (ambient, diffuse, emissive, specular)</p> <p>Shader-Parameter: Transparenz, Glanz und Beleuchtungsmodell</p> <p>Texturzuordnungen</p>
BEMERKUNG	<p>Materialien können mehrere Texturen und Eigenschaften kombinieren, um realistische visuelle Effekte zu erzeugen.</p> <p>Die Klasse ist final und kann nicht weiter abgeleitet werden.</p> <p>Die Ressource verwaltet eine Matrix (matrix_colors), die Farben speichert und überprüfbar ist.</p>
BEISPIELE	<p>Ein Material mit den folgenden Eigenschaften:</p> <ul style="list-style-type: none"> • Name: "BrickMaterial" • Diffuse Farbe: (0.8, 0.2, 0.2) • Specular Farbe: (1.0, 1.0, 1.0) • Textur: brick_diffuse.png

BEGRIFF	Object_Resource
BESCHREIBUNG	<p>Eine Object_Resource repräsentiert die geometrischen und materiellen Eigenschaften eines 3D-Objekts, einschließlich seiner Vertices, Indices und zugehörigen Metadaten. Sie dient als zentrale Datenstruktur, um Objekte für die Verwendung in Rendering zu speichern und zu verwalten.</p>
IST-EIN	<p>Die Object_Resource ist eine abgeleitete Klasse von Base_Resource, die speziell für die Verwaltung und Darstellung von 3D-Objekten konzipiert ist.</p>
KANN-SEIN	<p>Ein 3D-Objekt mit geometrischen Eigenschaften wie Position, Normalen, Texturkoordinaten und Farben.</p> <p>Eine Ressource, die Materialzuweisungen für Objekte speichert.</p> <p>Eine Datenstruktur für Indices und Vertices, die für Rendering oder andere Berechnungen verwendet werden.</p>

ASPEKT	<p>Die Object_Resource organisiert die grundlegenden Daten eines 3D-Objekts. Dies umfasst:</p> <p>Vertices: Enthalten Position, Farbe, Texturkoordinaten und Normalen. Indices: Beschreiben die Verbindung zwischen Vertices, um Dreiecke oder andere Primitive zu bilden. Materialinformationen: Speichern Materialnamen und Materialzuweisungen. Matrixrepräsentationen: Effiziente Speicherung und Prüfung von Vertices und Indices über Matrizen.</p>
BEMERKUNG	<p>Die Klasse enthält sowohl strukturierte Vertices als auch Indices, die die Grundfigur eines Objekts beschreiben. Sie unterstützt die effiziente Speicherung und den Zugriff auf diese Daten durch Matrizen.</p>
BEISPIELE	<p>Ein 3D-Modell einer Figur: Vertices: Enthalten Positionen, Texturkoordinaten und Farben für die Geometrie. Indices: Definieren die Dreiecke des Modells. Material: Zuweisung eines Materials mit Diffuse- und Normalmaps.</p>
BEGRIFF	Base Resource
BESCHREIBUNG	<p>Die Base_Resource-Klasse stellt eine abstrakte Basisklasse dar, die als Grundlage für verschiedene Ressourcentypen dient. Sie definiert grundlegende Eigenschaften wie eine eindeutige UUID, den Ressourcentyp (z. B. Material oder Objekt) und den Pfad zur Ressource.</p>
IST-EIN	<p>Die Base_Resource ist eine abstrakte Basisklasse, von der spezialisierte Ressourcenklassen wie Material_Resource und Object_Resource abgeleitet werden.</p>
KANN-SEIN	<p>Eine Material-Ressource (Typ MATERIAL). Eine Objekt-Ressource (Typ OBJECT). Jede weitere spezialisierte Ressource, die von dieser Klasse abgeleitet wird.</p>
ASPEKT	<p>Die Klasse Base_Resource definiert grundlegende Attribute und Funktionen für Ressourcen:</p> <p>UUID: Eine eindeutige Identifikation der</p>

	<p>Ressource.</p> <p>Typ: Der Ressourcentyp, der durch das ResourceType-Enum festgelegt ist.</p> <p>Pfad: Der Speicherpfad der Ressource.</p>
BEMERKUNG	<p>Die Klasse ist abstrakt und wird nicht direkt instanziiert.</p> <p>Sie bietet grundlegende Getter- und Setter-Methoden, um auf ihre Attribute zuzugreifen und sie zu bearbeiten.</p> <p>Die UUID sorgt für eine eindeutige Identifizierung, während der Ressourcentyp und der Pfad spezifische Eigenschaften beschreiben.</p>
BEISPIELE	<p>Material-Ressource:</p> <ul style="list-style-type: none"> • UUID: 123e4567-e89b-12d3-a456-426614174000 • Typ: MATERIAL • Pfad: /materials/brick.mtl <p>Objekt-Ressource:</p> <ul style="list-style-type: none"> • UUID: 789e0123-e45f-67a8-bcde-987654321000 • Typ: OBJECT • Pfad: /objects/car.obj

BEGRIFF	Raytracer
BESCHREIBUNG	Der Raytracer ist ein System, das mithilfe von SDL2 zur Darstellung realistischer 3D-Szenen arbeitet. Es nutzt Komponenten und Daten aus Ressourcen wie Materialien, Objekten und Lichtquellen, um die Szenen zu berechnen und auf den Bildschirm zu rendern. Dabei werden physikalisch basierte Berechnungen wie Lichtstrahlverfolgung und Materialeigenschaften verwendet, um realistische Effekte wie Reflexion, Brechung und Schatten zu erzeugen.
IST-EIN	Ein Raytracer, der Ressourcen aus einem Ressourcenmanagement-System bezieht (z. B. Material- und Objektressourcen). SDL2 zur Fenster- und Ereignisverwaltung verwendet.
KANN-SEIN	Ein Renderer, der: <p>Materialien aus Material_Resource liest, um Farben, Texturen und Beleuchtung zu definieren.</p> <p>Objektdaten aus Object_Resource nutzt, um „Dreiecken“ zu rendern.</p> <p>Lichteigenschaften berücksichtigt, um Beleuchtung und Schatten zu berechnen.</p> <p>Eine interaktive Simulation mit SDL2, die Benutzereingaben wie Kamerabewegungen oder Objektinteraktionen verarbeitet.</p>
ASPEKT	<p>Ressourcenverwaltung: Nutzt Daten aus Base_Resource-abgeleiteten Klassen (Material_Resource, Object_Resource) für die Szene.</p> <p>Physikalisch basierte Berechnungen: Berechneten Lichtstrahlen und ihre Interaktion mit Objekten in der Szene.</p>

BEMERKUNG

Der Raytracer verwendet SDL2 als Basis, um plattformübergreifend zu arbeiten. Die Komponenten der Szene werden aus einem zentralen Ressourcenmanagement-System bezogen, was die Organisation und Wiederverwendbarkeit erleichtert.

Materialien und Objekte werden physikalisch korrekt behandelt, um realistische Szeneneffekte wie Reflexion, Brechung und Schatten zu erzeugen.

Durch die Nutzung von SDL2 können Szenen in Echtzeit angepasst und Benutzereingaben dynamisch verarbeitet werden.

BEISPIELE



BEGRIFF	Material Importer
BESCHREIBUNG	Ein Modul, das Materialeigenschaften wie Farben, Texturen und Beleuchtungsparameter aus Dateien liest und diese für die Verwendung im Raytracer speichert.
IST-EIN	Eine Komponente des Raytracers, die Daten über Materialien aus externen Dateien extrahiert und in ein nutzbares Format umwandelt.
KANN-SEIN	Eine Schnittstelle zum Laden von Materialien aus .mtl-Dateien.
ASPEKT	Liest Materialinformationen wie Farben, Glanz oder Transparenz. Unterstützt Texturzuweisungen, die die Oberfläche eines Materials beschreiben. Stellt sicher, dass alle Materialien korrekt geladen und im System verfügbar sind.
BEMERKUNG	Der Material Importer überprüft die Konsistenz der Daten und stellt sicher, dass Texturpfade gültig sind. Er optimiert die Verarbeitung, indem redundante Daten ignoriert werden.
BEISPIELE	„newmtl Metal Ka 0.0 0.0 0.0 Kd 0.5 0.5 0.5 Ks 1.0 1.0 1.0 map_Kd metal_diffuse.png map_Bump metal_bump.png Textur“

BEGRIFF	Objekt Importer
BESCHREIBUNG	Ein Modul, das die Geometrie von Objekten (Formen, Strukturen) und deren Materialzuweisungen aus Dateien liest und sie für den Raytracer bereitstellt.
IST-EIN	Eine Komponente, die Daten über die Geometrie und Struktur von Objekten aus externen Dateien extrahiert und in ein nutzbares Format überträgt. In unserem Fall Dreiecke ausliest.
KANN-SEIN	Eine Schnittstelle zum Laden von Objektdaten aus .obj-Dateien.
ASPEKT	Liest die Geometrie eines Objekts (Positionen, Oberflächen, Struktur). Verknüpft Objekte mit Materialien, um realistische Darstellungen zu ermöglichen. Stellt sicher, dass alle Objekte korrekt geladen und miteinander kompatibel sind.

BEMERKUNG	Der Importer überprüft, ob alle Materialien, die einem Objekt zugeordnet sind, existieren.
BEISPIELE	Er entfernt unnötige oder doppelte Daten, um die Performance zu verbessern. „mtllib teapotonchess.mtl v 43.967953 129.156097 107.249275 v 43.372643 131.406357 107.249275 v 44.063202 132.156448 107.249275“ @ Aus der teapotonchess-Datei

BEGRIFF	Fragment Shader
BESCHREIBUNG	Der Fragment Shader ist ein wesentlicher Bestandteil der Rendering-Pipeline und wird verwendet, um die endgültige Farbe eines Pixels zu berechnen.
IST-EIN	Ein GLSL-Shader (OpenGL Shading Language), der in der Pipeline der Grafikkarte ausgeführt wird, um Farben und andere Pixelattribute zu berechnen.
KANN-SEIN	Ein einfacher Shader, der: Die vom Vertex-Shader interpolierte Farbe (frag_color) direkt übernimmt. Die berechnete Farbe als Ausgabe (color) an den Framebuffer schreibt.
ASPEKT	Shader Version: Verwendet GLSL-Version 330 („#version 330 core“). Input: frag_color (eine interpolierte Farbe, die vom vorherigen Shader übergeben wird). Output: color (die endgültige Pixel-Farbe, die auf dem Bildschirm erscheint).
BEMERKUNG	Der Shader ist minimal und übernimmt lediglich die interpolierte Fragmentfarbe. Erweiterungen könnten Texturen, Beleuchtungseffekte oder andere Farbmanipulationen einfügen. Die Struktur und Syntax sind auf moderne OpenGL-Standards ausgelegt.

BEISPIELE

Eingabe: frag_color = vec4(1.0, 0.0, 0.0, 1.0) (Rot).
Ausgabe: color = vec4(1.0, 0.0, 0.0, 1.0) (Pixel wird rot gefärbt).

BEGRIFF

Vertex Shader

BESCHREIBUNG

Der Vertex Shader verarbeitet einzelne Vertices eines 3D-Objekts und berechnet ihre Position sowie weitere Attribute für die nächste Stufe der Rendering-Pipeline.

IST-EIN

Ein GLSL-Shader (OpenGL Shading Language), der die Eingabedaten eines 3D-Objekts (Position und Farbe) verarbeitet und für die nächste Stufe der Pipeline vorbereitet.

KANN-SEIN

Ein Vertex-Shader bearbeitet Vertex-Positionen und transformiert diese, um sie in den MVP-Raum zu projizieren, und Farben interpoliert, die anschließend an den Fragment-Shader übergeben werden.

ASPEKT

Shader Version: Verwendet GLSL-Version 330 (#version 330 core).
Inputs:
position (Vertex-Position als vec3).
color (Vertex-Farbe als vec3).
Uniform:
modelViewProj (eine Transformationsmatrix, die das Modell in den Clip-Space transformiert).
Outputs:
frag_color (die interpolierte Vertex-Farbe, die an den Fragment-Shader übergeben wird).

BEMERKUNG

Der Shader verwendet eine Transformationsmatrix (modelViewProj), die die Modell-, View- und Projektionsmatrizen kombiniert.

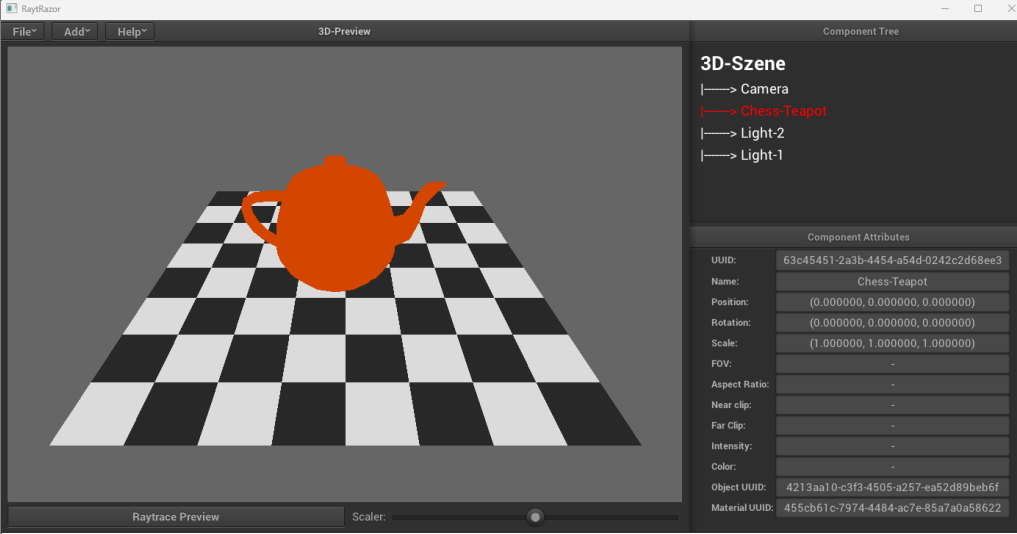
BEISPIELE

Eingabe: position = vec3(1.0, 1.0, 1.0)
Transformationsmatrix: modelViewProj = mat4(...)
Ausgabe: gl_Position = modelViewProj * vec4(1.0, 1.0, 1.0, 1.0)

BEGRIFF	Ressourcen-Ordner
BESCHREIBUNG	<p>Der Ressourcen-Ordner dient als zentrale Ablage für alle benötigten Daten des Raytracers. Er enthält Unterordner, die spezifische Arten von Ressourcen speichern:</p> <p>assets: Enthält .obj- und .mtl-Dateien, die Objekte und Materialien definieren.</p> <p>scenes: Beinhaltet vorgefertigte Szenen, die aus Objekten und Materialien bestehen und direkt im Raytracer angezeigt werden können.</p>
IST-EIN	Eine strukturierte Sammlung von Ressourcen, die für den Betrieb des Raytracers erforderlich sind. Sie enthält Rohdaten (Objekte und Materialien) und fertige Szenen zur Demonstration.
KANN-SEIN	<p>assets: Ein Unterordner, der:</p> <ul style="list-style-type: none"> .obj-Dateien speichert, die die Geometrie von 3D-Objekten beschreiben. .mtl-Dateien enthält, die Materialdefinitionen wie Farben, Texturen und Beleuchtungsparameter für die Objekte bereitstellen. <p>scenes: Ein Unterordner, der:</p> <ul style="list-style-type: none"> Szenendateien speichert, die mehrere Objekte, Materialien und ihre Anordnung in einer 3D-Welt definieren.
ASPEKT	<p>Assets:</p> <p>.obj-Dateien enthalten:</p> <ul style="list-style-type: none"> Vertices (Positionen, Normalen, Texturkoordinaten). Indices zur Definition von Dreiecken. Materialzuweisungen (usemtl), die auf .mtl-Dateien verweisen. <p>.mtl-Dateien enthalten:</p> <ul style="list-style-type: none"> Farben (ambient, diffuse, specular). Beleuchtungseigenschaften (Shininess, Transparenz). Texturpfade (z. B. map_Kd für Diffuse-Texturen). <p>Scenes:</p> <p>Fertige Szenen mit definierten Objekten, Lichtquellen und Kamerapositionen.</p> <p>Typischerweise JSON- oder XML-basierte Dateien zur Konfiguration.</p>

BEMERKUNG	<p>Der Ordner ist modular aufgebaut, sodass Assets und Szenen leicht hinzugefügt oder geändert werden können.</p> <p>Szenen im scenes-Ordner nutzen die Daten aus dem assets-Ordner und kombinieren sie zu kompletten Darstellungen.</p> <p>Der Zugriff auf die Ressourcen erfolgt über die entsprechenden Importer (Material Importer und Objekt Importer).</p>
BEISPIELE	<pre> ressources/ ├── assets/ │ ├── cube.obj │ ├── cube.mtl │ ├── sphere.obj │ ├── sphere.mtl │ └── textures/ │ ├── brick_diffuse.png │ └── metal_specular.png └── scenes/ ├── scene1.json └── scene2.json </pre>

BEGRIFF	Main Szene
BESCHREIBUNG	<p>Die Main-Szene ist die zentrale Klasse des 3D-Previews und Raytracers. Sie verwaltet alle Komponenten und Ressourcen einer Szene und bietet Funktionen zur Anzeige, Interaktion und Modifikation. Die Klasse nutzt eine grafische Oberfläche, die mit NanoGUI erstellt wurde, sowie OpenGL zur Darstellung der 3D-Szene. Sie integriert zudem eine JSON-basierte Importfunktion für Szenen und Ressourcen.</p>
IST-EIN	<p>Eine Klasse, die:</p> <ul style="list-style-type: none"> Komponenten (z. B. Kamera, Licht, Render-Objekte) und Ressourcen (z. B. Materialien, Meshes) verwaltet. Die grafische Benutzeroberfläche organisiert und Benutzereingaben verarbeitet.
KANN-SEIN	<p>3D-Szenenmanager: Organisiert die Darstellung von Objekten, Licht und Kameras in einer Szene.</p> <p>Komponenten- und Ressourcenmanager: Verknüpft Ressourcen mit Komponenten und ermöglicht Änderungen in der GUI.</p> <p>GUI-Controller: Stellt Fenster für die Vorschau, Komponentenliste und Attributbearbeitung bereit.</p>
ASPEKT	<p>Komponentenverwaltung:</p> <ul style="list-style-type: none"> Verwalten von Kameras, Lichtern und Render-Objekten über UUIDs. Automatisches Hinzufügen einer Kamera, falls keine existiert. <p>Ressourcenverwaltung:</p> <ul style="list-style-type: none"> Laden und Verknüpfen von Materialien und Objektressourcen. JSON-Import von Szenen zur dynamischen Laufzeit. <p>3D-Vorschau:</p> <ul style="list-style-type: none"> Rendering der Szene in einem OpenGL-Canvas (Preview_Canvas). Unterstützung für Kamerabewegungen (WASD-Steuerung).

	<p>GUI-Funktionen:</p> <p>Attributfenster für Komponenten.</p> <p>Interaktive Baumansicht der Szene (TreeView_Widget).</p> <p>Bedienelemente wie Schieberegler, Buttons und Menüleisten.</p>
BEMERKUNG	<p>Die Klasse ist als Singleton (instance) implementiert, um globale Zugriffspunkte für Szenenverwaltung und GUI zu ermöglichen.</p> <p>Das Rendering verwendet OpenGL und GLM, um Kameraperspektiven, Objekttransformationen und Beleuchtungen zu berechnen.</p> <p>Die Integration von JSON ermöglicht das Importieren und Speichern von Szenen in einem leicht lesbaren Format.</p> <p>Threading: Die Raytracer-Vorschau wird in einem separaten Thread ausgeführt, um die GUI responsiv zu halten.</p>
BEISPIELE	

BEGRIFF	Custom Label
BESCHREIBUNG	<p>Die Custom_Label-Klasse erweitert die Funktionalität von NanoGUI Labels, um eine benutzerdefinierte Beschriftung bereitzustellen. Sie ermöglicht die Anzeige von Text mit anpassbarer Schriftart, Farbe und Größe und unterstützt zusätzliche Funktionen wie Click-Events.</p>
IST-EIN	<p>Interaktionen durch Mausereignisse unterstützt.</p>
KANN-SEIN	<p>Ein statisches Label, das einfach nur Text anzeigt.</p> <p>Ein interaktives Label, das auf Benutzeraktionen (z. B. Klicks) reagiert.</p> <p>Eine visuelle Komponente in einem GUI, die Textinhalte dynamisch ändert.</p>
ASPEKT	<p>Erweitert die Basisfunktionalitäten mit:</p> <p>Unterstützt Click-Events durch einen Callback, der mit setCallback definiert wird.</p>

BEMERKUNG

Die Custom_Label-Klasse integriert sich nahtlos in das NanoGUI-Framework. Neben der visuellen Darstellung kann das Label interaktive Funktionen übernehmen (z. B. ein Button-Ersatz).

BEISPIELE

Den Text nach dem interagieren mit ihm die Farben wechseln zu lassen.

BEGRIFF	Component-Attributes-Widget
BESCHREIBUNG	Das ComponentAttributes_Widget ist eine spezialisierte selbsterstellte Widget-Klasse, die in NanoGUI verwendet wird, um die Attribute von Komponenten visuell darzustellen und zu bearbeiten. Es bietet Funktionen zum Anzeigen, Aktualisieren und Löschen von Komponentenattributen, die mit der 3D-Szene verknüpft sind.
IST-EIN	<p>Eine grafische Benutzeroberfläche (GUI), die:</p> <p>Die Attribute von Komponenten wie Camera_Component, Render_Component und Light_Component dynamisch anzeigt. Als Widget in ein übergeordnetes Fenster integriert wird.</p> <p>Die Interaktion und Bearbeitung von Attributen der 3D-Komponenten ermöglicht.</p>
KANN-SEIN	<p>Eine Attributanzeige: Zeigt aktuelle Werte wie Position, Rotation und Skalierung an.</p> <p>Ein Attributeditor: Ermöglicht dem Benutzer, Werte direkt zu bearbeiten (z. B. Kameraposition, Lichtintensität).</p> <p>Ein dynamisches Werkzeug: Reagiert auf Änderungen an der Komponente und aktualisiert die Anzeige automatisch.</p>
ASPEKT	Das ComponentAttributes_Widget generiert dynamische Widgets zur Bearbeitung, bietet Funktionen zum Anzeigen, Aktualisieren, Bearbeiten und Löschen sowie Hilfsfunktionen zur Umwandlung und Darstellung von Werten.
BEMERKUNG	<p>Das Widget unterstützt verschiedene Komponententypen (Base_Component, Camera_Component, Render_Component, Light_Component) und ist flexibel erweiterbar.</p> <p>Die dynamischen Widgets werden zur Laufzeit erstellt und ermöglichen es, Attribute flexibel darzustellen und zu bearbeiten.</p>

BEISPIELE

Component Attributes			
UUID:	63c45451-2a3b-4454-a54d-0242c2d68ee3		
Name:	Chess-Teapot		
Position:	0.000000	0.000000	0.000000
Rotation:	0.000000	0.000000	0.000000
Scale:	1.000000	1.000000	1.000000
FOV:	-		
Aspect Ratio:	-		
Near clip:	-		
Far Clip:	-		
Intensity:	-		
Color:	-	-	-
Object UUID:	4213aa10-c3f3-4505-a257-ea52d89beb6f		
Material UUID:	455cb61c-7974-4484-ac7e-85a7a0a58622		

BEGRIFF

BESCHREIBUNG

Component-Tree-Widget

Das TreeView-Widget ist ein GUI-Widget, das eine hierarchische Baumstruktur darstellt. Es ermöglicht das Hinzufügen, Entfernen und Aktualisieren von Knoten, wodurch die Organisation und Navigation von Komponenten in einer Szene erleichtert wird.

IST-EIN

Ein spezialisierter Widget-Container aus NanoGUI, der:


Eine scrollbare Baumstruktur zur Anzeige hierarchischer Beziehungen bietet. Mit einem ComponentAttributes_Widget verknüpft ist, um die Attribute eines ausgewählten Knotens anzuzeigen.

KANN-SEIN

Eine visuelle Darstellung der Komponentenhierarchie in einer Szene. Ein interaktives Werkzeug, das Benutzern ermöglicht:
Knoten (z. B. Objekte, Kameras, Lichter) hinzuzufügen.
Den Fokus auf einen bestimmten Knoten zu legen und dessen Attribute anzuzeigen.
Die Struktur durch Löschen oder Aktualisieren anzupassen.

ASPEKT

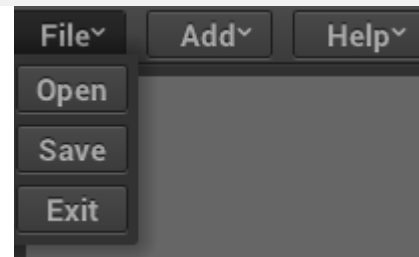
Knoten können hierarchisch organisiert und interaktiv ausgewählt werden, das Widget ist mit einem Attribut-Editor verknüpft, unterstützt die Navigation großer Strukturen und ermöglicht das Aktualisieren oder Zurücksetzen der Baumdarstellung.

BEMERKUNG	Die Baumstruktur ist flexibel, dynamisch, passt sich den Szenekomponenten an, verknüpft Knoten effizient mit Widgets und ermöglicht eine komfortable Navigation großer Strukturen.
BEISPIELE	 <pre> Component Tree 3D-Szene ----> Camera ----> Light_Added ----> Chess-Teapot ----> Light_Added ----> Light-2 ----> Light-1 </pre>

BEGRIFF	MenuBar-Widget
BESCHREIBUNG	Das MenuBar-Widget ist eine spezialisierte GUI-Komponente in NanoGUI, die eine Menüleiste bereitstellt. Es ermöglicht die Organisation von Dropdown-Menüs mit Optionen, die durch Benutzeraktionen ausgelöst werden können.
IST-EIN	Eine Menüleiste mit Hauptmenüs und zugehörigen Dropdown-Optionen bietet. Benutzeraktionen über definierte Callbacks für jede Menüoption ermöglicht.
KANN-SEIN	<p>Eine Steuerzentrale: Organisiert Funktionen wie "Datei öffnen", "Komponente hinzufügen" oder "Hilfe anzeigen".</p> <p>Reagiert auf Benutzereingaben und führt zugewiesene Aktionen aus.</p> <p>Ein dynamisches Menü: Menüs mit Dropdown-Optionen bieten einfache Navigation, verknüpfen Optionen mit Callbacks, unterstützen Aktionen wie Datei- und Verzeichniszugriff sowie Szenenverwaltung und erleichtern den Zugriff auf zentrale Funktionen.</p>
ASPEKT	
BEMERKUNG	Die Menüleiste kann flexibel an verschiedene Anforderungen angepasst werden, indem Menüs und Optionen dynamisch hinzugefügt werden. Dateipfade und Verzeichnisse werden validiert, um Benutzerfehler zu minimieren. Die Organisation und Darstellung der Menüs erfolgten in einer übersichtlichen

Dropdown-Struktur.

BEISPIELE



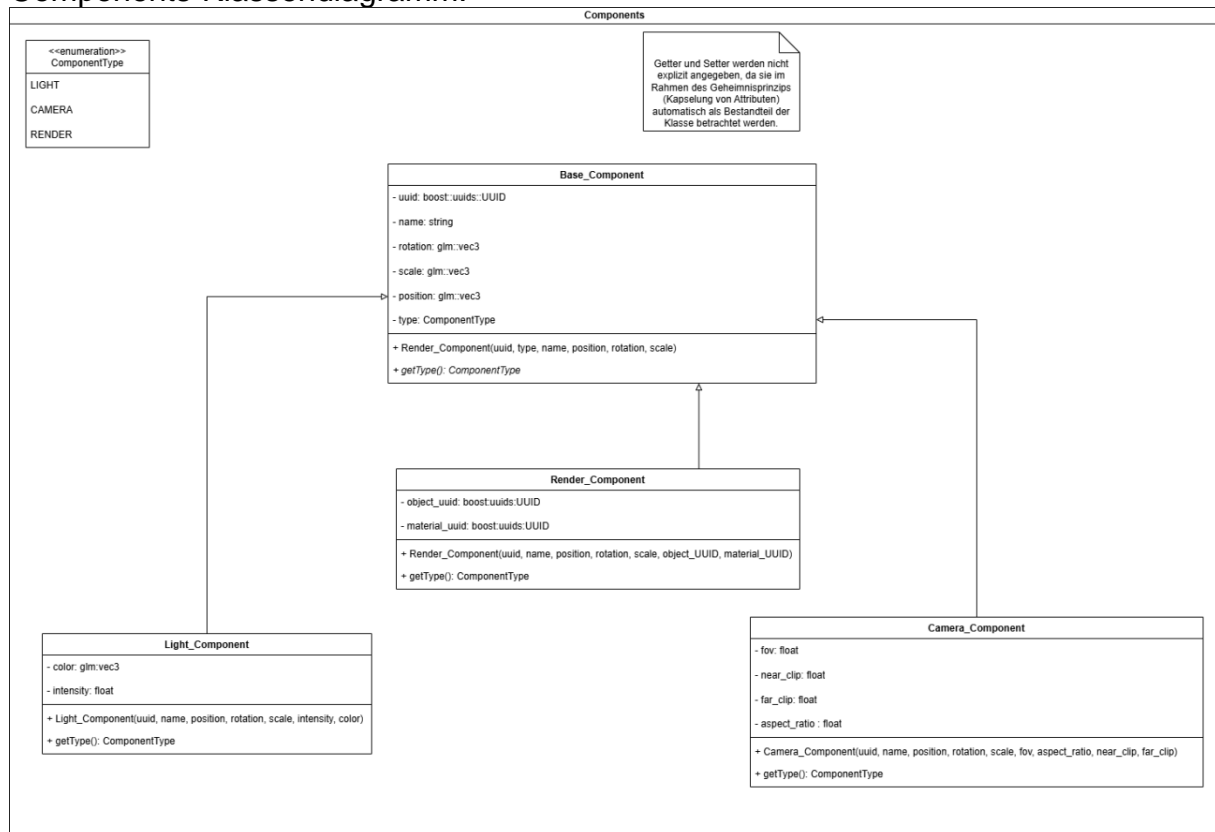
3. Domänenmodell

Das Domänenmodell des Raytracers wurde in verschiedene Module unterteilt, um die Aufgaben klar zu trennen und die Entwicklung effizient zu strukturieren. Die GUI (Grafische Benutzeroberfläche) ermöglicht die Darstellung der Szene und bietet dem Benutzer Interaktionsmöglichkeiten, wie die Steuerung der Kamera, die Anpassung von Attributen sowie die Verwaltung der Baumstruktur. Das Raytracing-Modul übernimmt als Kernkomponente die physikalisch basierte Berechnung von Effekten wie Reflexion, Brechung und Schatten, um realistische Bilder basierend auf den Szenendaten zu erstellen.

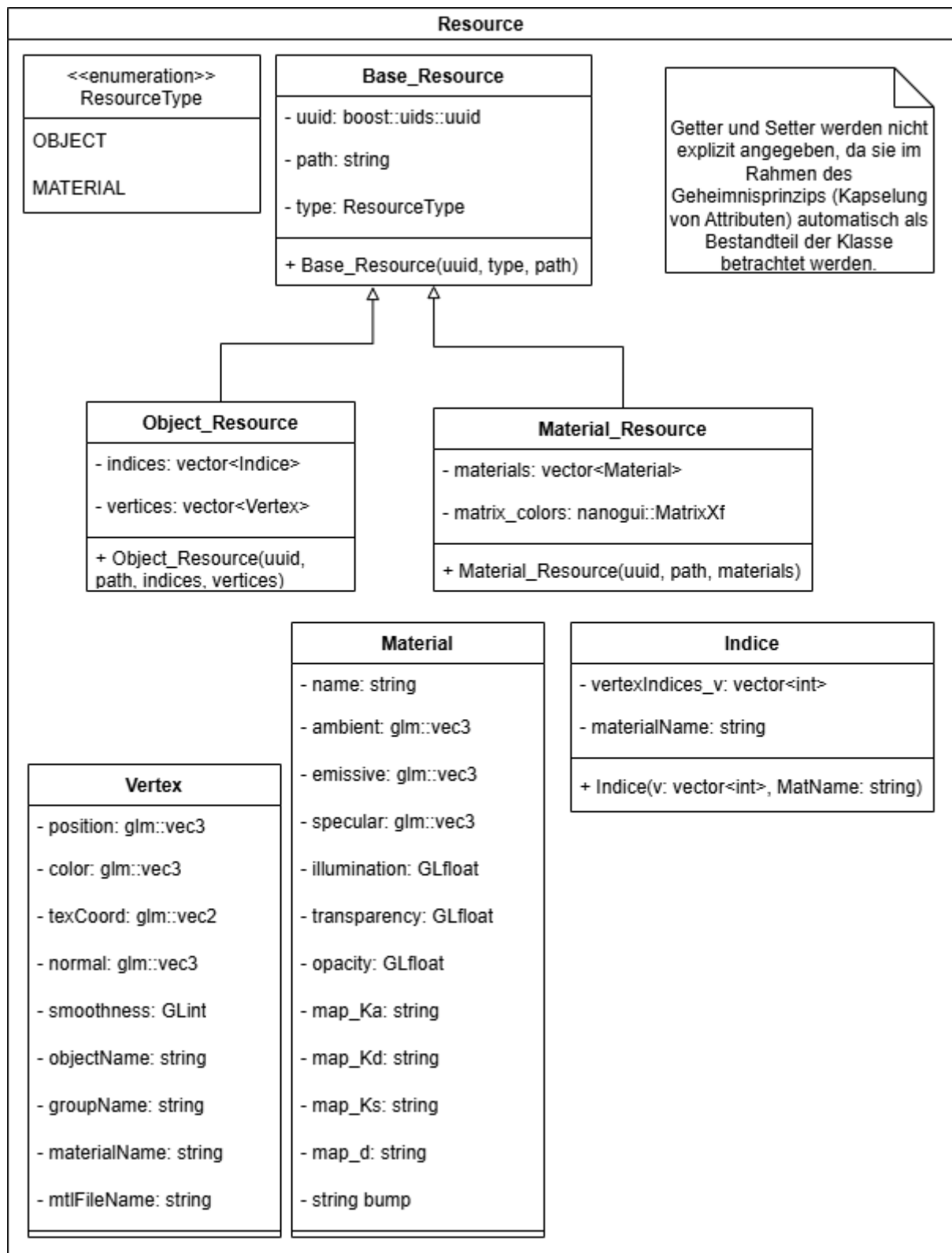
Der Importer dient dazu, Szenen, Objekte und Materialien aus externen Dateien zu laden und in ein für den Raytracer verständliches Format zu übersetzen, wodurch die Nutzung standardisierter OBJ/MTL-Files gewährleistet wird. Ergänzend dazu überprüft das Parsing-Modul die Konsistenz und Vollständigkeit der importierten Szenen, dass alle Daten korrekt verarbeitet werden können.

Das Converter-Modul übernimmt die Umwandlung von Daten zwischen verschiedenen Formaten und stellt die Kompatibilität zwischen den eingesetzten Modulen und Bibliotheken sicher. Schließlich verwalten die einzelnen Komponenten, wie Kamera, Lichtquellen und Materialien, die grundlegenden Bausteine der Szene und stellen deren Attribute für das Rendering und die Interaktion bereit. Diese klare Einteilung sorgt für eine modulare und erweiterbare Architektur, die eine optimale Zusammenarbeit der verschiedenen Module ermöglicht.

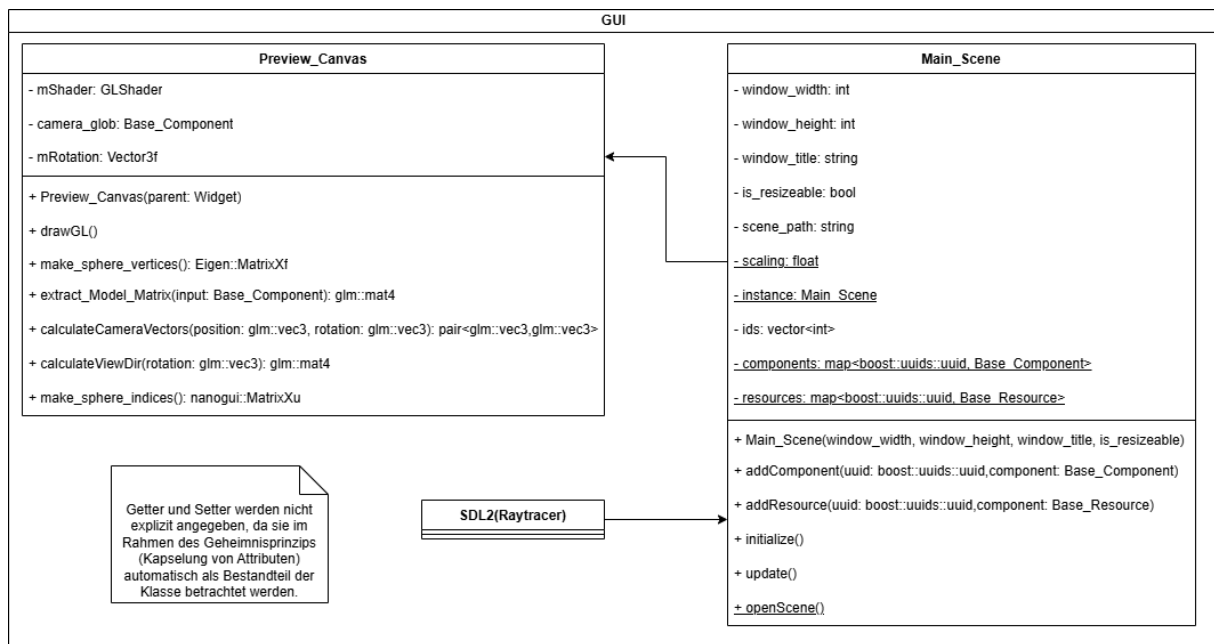
Components-Klassendiagramm:



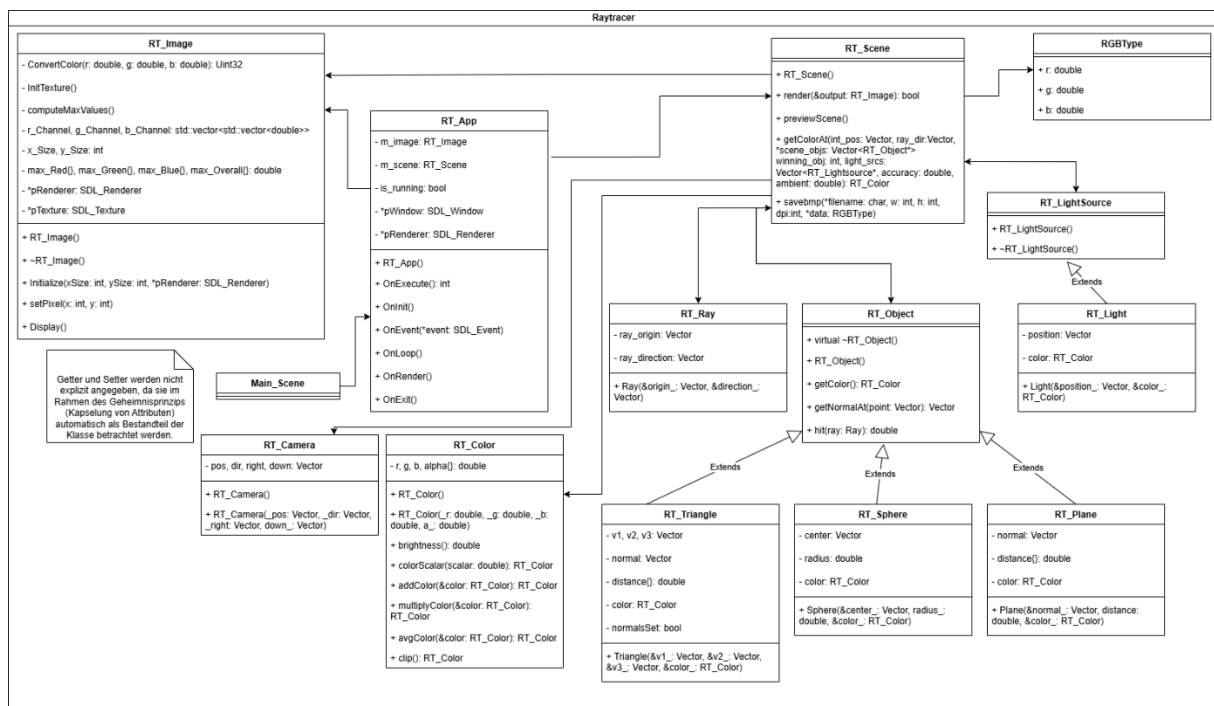
Resource-Klassendigramm:



GUI-Klassendiagramm:



Raytracer-Klassendiagramm:



4. Anforderungsdefinition

AKTEUR	AUFTRAGGEBER
BESCHREIBUNG	Institut für Softwaretechnik
ROLLE	Der Auftraggeber stellt die Anforderungen bereit, über die das Produkt am Ende verfügen muss. Nach der Entwicklung testet er das Resultat und entscheidet darüber, ob das fertige Produkt die gestellten Anforderungen erfüllt.
AKTEUR	DEVELOPER
BESCHREIBUNG	RayForge (wir)
ROLLE	Sind verantwortlich für das Erstellen des Produkts und müssen dabei auf die Wünsche und Anforderungen des Auftraggebers eingehen.
AKTEUR	CLIENT
BESCHREIBUNG	Die Software des Teilnehmers
ROLLE	Einlesen von aus Dreiecken zusammengesetzten Oberflächenmodellen aus Dateien im .obj/.mtl-Format.
AKTEUR	TEILNEHMER
BESCHREIBUNG	Endnutzer des Projekts
ROLLE	Personenkreis, der die Software nutzt, um „photorealistische“ Darstellungen von Dreiecksszenen zu generieren.
AKTEUR	STACKEHOLDER
BESCHREIBUNG	Prof. Dr. R. Lunde
ROLLE	Stakeholder sind Teilhaber des Projekts. Sie haben außerdem berechtigtes Interesse am Verlauf und dem Ergebnis.

4.1 Funktionale Anforderungen

ID	FA1
TITEL:	Einlesen von .obj/.mtl-Dateien
BESCHREIBUNG	Der Raytracer muss in der Lage sein, Oberflächenmodelle, die aus Dreiecken bestehen, aus Dateien im .obj- und .mtl-Format einzulesen.
BEGRÜNDUNG	Dies ist notwendig, um geometrische Objekte für das Rendering bereitzustellen. Die .obj- und .mtl-Formate sind weit verbreitet und ermöglichen eine einfache Integration von 3D-Modellen.
ABHÄNGIGKEITEN	-

ID	FA2
TITEL:	Freie Wahl der Sicht auf das Objekt
BESCHREIBUNG	Der Raytracer muss es ermöglichen, die Lage der Projektionsebene, den Bildausschnitt und das Projektionszentrum flexibel zu wählen. Zudem muss ein einheitliches Dateiformat für sichtrelevante Daten definiert werden.
BEGRÜNDUNG	Diese Anforderung ist wichtig, um eine benutzerdefinierte Ansicht des Modells zu ermöglichen. Eine gemeinsame Dateiformatspezifikation fördert die Zusammenarbeit zwischen den Gruppen und stellt sicher, dass alle Sichtdaten konsistent gespeichert werden.
ABHÄNGIGKEITEN	FA1

ID	FA3
TITEL:	3D-Preview für Ansichts- und Beleuchtungsdefinition
BESCHREIBUNG	Der Raytracer muss eine Echtzeit-Visualisierung der darzustellenden Objekte, Kamera und Lichtquellen bieten. Die 3D-Preview soll eine schnelle Reaktion auf Änderungen der Parameter ermöglichen und eine detaillierte Darstellung der Objekte (einschließlich Texturen) unterstützen
BEGRÜNDUNG	Eine Echtzeit-Visualisierung ermöglicht es Benutzern, Anpassungen sofort zu sehen, was die Qualität der Ergebnisse verbessert. Die Möglichkeit, mit Lichtquellen und Kameraeinstellungen zu experimentieren, ist entscheidend für ein effektives Design.
ABHÄNGIGKEITEN	FA2

ID	FA4
TITEL:	Bildberechnung durch Anwendung von Raytracing
BESCHREIBUNG	Der Raytracer muss die Bildberechnung durch Strahlverfolgung vom Projektionszentrum in Richtung der Projektionsebene realisieren.
BEGRÜNDUNG	Die Anwendung von Raytracing ermöglicht realistische Bilddarstellungen durch physikalisch basierte Lichtberechnung und effektive Simulation von Reflexion und Brechung. Dies ist entscheidend für die Qualität und Genauigkeit der erzeugten Bilder.
ABHÄNGIGKEITEN	FA1, FA3, FA2

ID	FA5
TITEL:	Bilddarstellung auf dem Monitor sowie Speicherung in einem gängigen Grafikformat
BESCHREIBUNG	Der Raytracer muss in der Lage sein, die berechneten Bilder auf dem Monitor darzustellen und sie in einem gängigen Grafikformat (z. B. PNG, JPEG, BMP) zu speichern.
BEGRÜNDUNG	Diese Anforderung ist entscheidend, um die Benutzererfahrung zu verbessern und die Interoperabilität mit anderen Softwareanwendungen zu gewährleisten. Eine unabhängige Bilddarstellung und -speicherung ist wichtig, um Flexibilität und Zukunftssicherheit zu gewährleisten.
ABHÄNGIGKEITEN	FA4, FA1, FA2

ID	FA6
TITEL:	Hinzufügen von Objekten oder Lichtquellen in der aktuellen Szene
BESCHREIBUNG	Durch einen Buttonaufruf können Objekte (.obj/.mtl) oder Lichtquellen, dynamisch zu einer bestehenden Szene hinzugefügt werden.
BEGRÜNDUNG	Die Szenen werden dadurch dynamischer und der Endnutzer hat mehr Freiheit sich seine eigene Szenen zu gestalten.
ABHÄNGIGKEITEN	FA1, FA2, FA3

ID	FA7
TITEL:	Löschen von Objekten oder Lichtquellen in der aktuellen Szene
BESCHREIBUNG	Durch einen Buttonaufruf können Objekte (.obj/.mtl) oder Lichtquellen, dynamisch zu einer bestehenden Szene entfernt werden.
BEGRÜNDUNG	Die Szenen werden dadurch dynamischer und der Endnutzer hat mehr Freiheit sich seine eigene Szenen zu gestalten.
ABHÄNGIGKEITEN	FA1, FA2, FA3

ID	FA8
TITEL:	Short-Cuts für wichtige Funktionen
BESCHREIBUNG	Implementierung von Short-Cuts für essenzielle Funktionen wie: Szenen rendern, „Szene öffnen“ ausführen, Lichtelemente hinzufügen, Programm schließen
BEGRÜNDUNG	Durch die Einführung von Short-Cuts werden Szenen dynamischer und benutzerfreundlicher. Dies ermöglicht Endnutzern mehr Freiheit und Effizienz bei der Gestaltung eigener Szenen.
ABHÄNGIGKEITEN	FA1, FA2

ID	FA9
TITEL:	Kamera per (WASDEFRRF) bewegen
BESCHREIBUNG	W: nach vorne A: nach links S: zurück D: nach rechts E: nach oben F: nach unten R/F: Neigung ändern
BEGRÜNDUNG	Durch die Einführung von Short-Cuts und Kamera-Steuerung werden Szenen dynamischer und benutzerfreundlicher. Dies ermöglicht Endnutzern mehr Freiheit und Effizienz bei der Gestaltung eigener Szenen.
ABHÄNGIGKEITEN	FA2

4.2 Nicht funktionale Anforderungen

ID	QA1
TITEL:	Robustheit/ Zuverlässigkeit
BESCHREIBUNG	Die Anwendung darf nicht abstürzen. Bei 100 Instanzen darf die Anwendung maximal 1 aufgrund eines Fehlers abstürzen.
BEGRÜNDUNG	Ein möglichst fehlerfreies Spiel ist für eine nutzerfreundliche Erfahrung zwingend notwendig.

ID	QA2
TITEL:	Effizienz der Bildberechnung
BESCHREIBUNG	Die Bildberechnung sollte möglichst schnell erfolgen, wobei der Speicherverbrauch bei der Umsetzung von Beschleunigungsstrategien nicht vernachlässigt werden darf.
BEGRÜNDUNG	Eine effiziente Bildberechnung ist entscheidend für die Benutzererfahrung und die Anwendbarkeit des Raytracers in realistischen Szenarien.

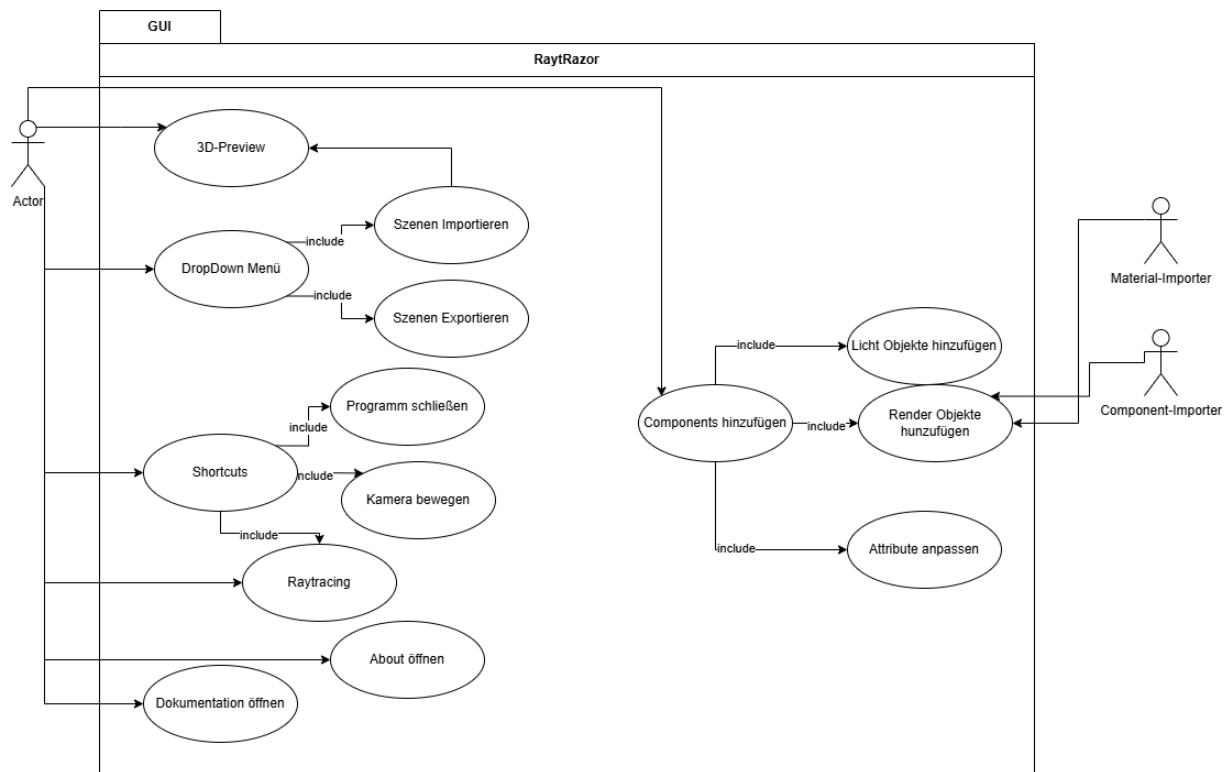
ID	QA3
TITEL:	Wartbarkeit der Software
BESCHREIBUNG	Die Software soll so gestaltet sein, dass sie von einem neuen Team zu einem späteren Zeitpunkt weiterentwickelt werden kann.
BEGRÜNDUNG	Eine wartbare Softwarearchitektur ist entscheidend, um die zukünftige Entwicklung und Erweiterung der Anwendung zu ermöglichen.

ID	QA4
TITEL:	Anwendungssprache, Implementierungssprache und Dokumentationssprache
BESCHREIBUNG	<ul style="list-style-type: none"> • Die Benutzerschnittstelle der Anwendung (d.h. aller Komponenten) kann auf Deutsch oder Englisch gestaltet werden. • Die Implementierungssprache (Bezeichner im Source Code und Kommentare) soll jedoch Englisch sein. • Sonstige Dokumente wie Benutzerhandbuch, Testdokumentationen, Projekttagbuch, usw. können auf Deutsch oder Englisch verfasst werden.

ID	QA5
TITEL:	Programmiersprachen und Technologien
BESCHREIBUNG	Als Programmiersprache sollten Java, JavaScript, C# und C++ genutzt werden

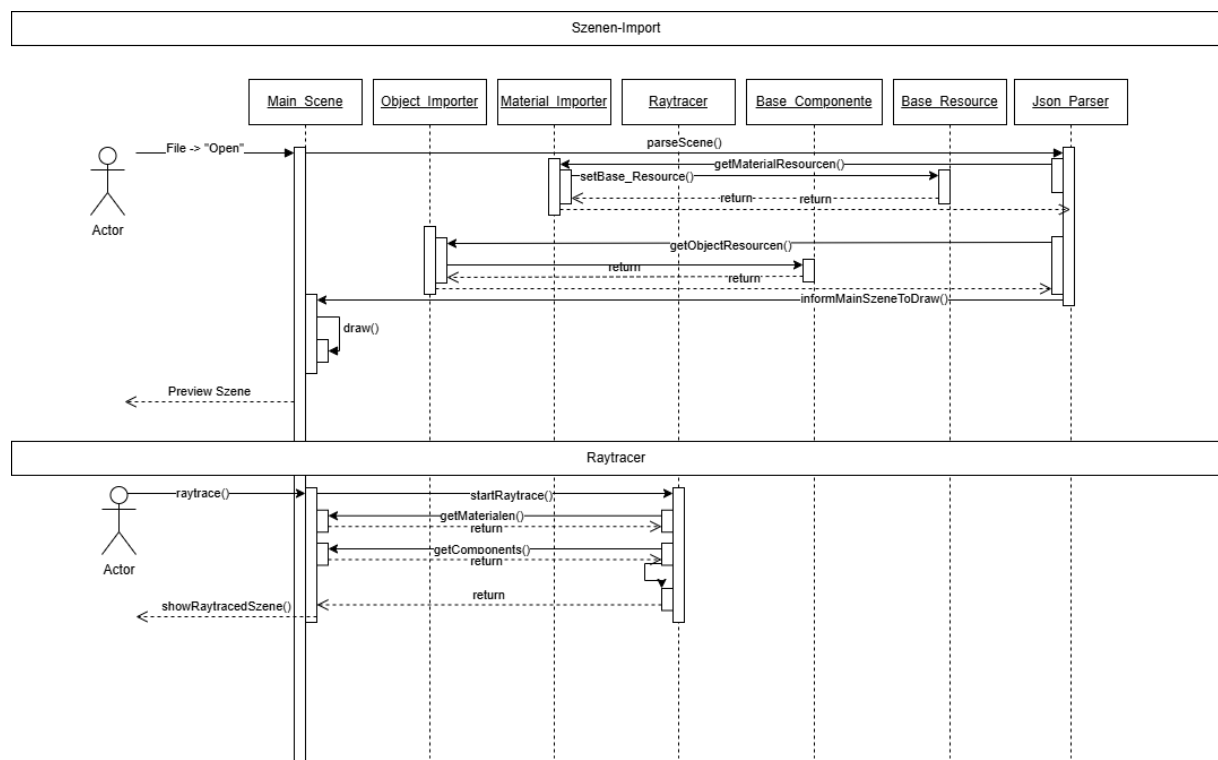
ID	QA6
TITEL:	Plattformen
BESCHREIBUNG	Der Benutzer-Client und Editor müssen entweder auf einer aktuellen Linux-Distribution, aktuellen Windows Version oder basierend auf Webtechnologien, in einem aktuellen, standartkonformen Browser Laufen.
BEGRÜNDUNG	Ermöglichen Nutzern der jeweiligen Plattform(en) den Raytracer zu nutzen.

5. Anwendungsfälle



6. Abläufe

Hierbei weiche ich vom Standard ab und tue nur die wichtigsten umsetzen!



7. Benutzerschnittstellen

7.1 3D – Szene JSON

```
{
  "renderentity": [
    {
      "name": "Chess-Teapot",
      "uuid": "63c45451-2a3b-4454-a54d-0242c2d68ee3",
      "Translation": {
        "position": {
          "x": 0,
          "y": 0,
          "z": 0
        },
        "rotation": {
          "x": 0,
          "y": 0,
          "z": 0
        },
        "scale": {
          "x": 1,
          "y": 1,
          "z": 1
        }
      },
      "components": {
        "RenderComponent": {
          "objUUID": "4213aa10-c3f3-4505-a257-ea52d89beb6f",
          "matUUID": "455cb61c-7974-4484-ac7e-85a7a0a58622"
        }
      }
    },
    {
      "name": "Light-1",
      "uuid": "b7d03d91-52ad-49b2-8c48-cd6a7e9ae822",
      "Translation": {
        "position": {
          "x": -250,
          "y": 250,
          "z": 0
        }
      },
      "components": {
        "LightComponent": {
          "intensity": 0.7,
          "color": {

```

```

        "r": 0,
        "g": 0,
        "b": 255
    }
}
},
{
    "name": "Light-2",
    "uuid": "abbea383-81e9-4f72-a351-a40a4cce4de8",
    "Translation": {
        "position": {
            "x": 250,
            "y": 250,
            "z": 0
        }
    },
    "components": {
        "LightComponent": {
            "intensity": 0.3,
            "color": {
                "r": 255,
                "g": 0,
                "b": 0
            }
        }
    }
}
],
"cameraentity": [
{
    "name": "Camera",
    "uuid": "163fe421-7c52-41f4-a2b8-7cb77f88566f",
    "Translation": {
        "position": {
            "x": 10,
            "y": 250,
            "z": 400
        }
    },
    "rotation": {
        "x": -35,
        "y": 0,
        "z": 0
    }
},
    "components": {
        "CameraComponent": {
            "fov": 60,
            "aspectRatio": 1.77,
            "nearClip": 0.1,
            "farClip": 10000
        }
    }
}
]

```

```

    }
  }
},
"resources": [
  {
    "uuid": "4213aa10-c3f3-4505-a257-ea52d89beb6f",
    "type": "obj",
    "path": "..\\assets\\miscellaneous\\miscellaneous\\teapot-
chess\\teapotonchess.obj"
  },
  {
    "uuid": "455cb61c-7974-4484-ac7e-85a7a0a58622",
    "type": "mat",
    "path": "..\\assets\\miscellaneous\\miscellaneous\\teapot-
chess\\teapotonchess.mtl"
  }
],
"metadata": {
  "backgroundColor": [0.1, 0.1, 0.1],
  "globalIllumination": true,
  "renderMode": "pathtracing",
  "maxDepth": 5,
  "samplesPerPixel": 100
}
}

```

In einer 3D-Szene beschreibt das für einen Raytracer verwendete JSON alle wesentlichen Szenenelemente und gliedert sie in zwei Hauptkomponenten: **Entities** und **Resources**.

Die Entities repräsentieren die verschiedenen Objekte in der Szene, wie z.B. 3D-Modelle oder Lichtquellen (Kamera). Jedes dieser Objekte hat bestimmte Eigenschaften, die seine Platzierung und Ausrichtung im Raum bestimmen. Dazu gehören Position, Rotation und Skalierung, die jeweils die räumliche Anordnung des Objekts bestimmen und beeinflussen, wie es im endgültigen Bild erscheint. Darüber hinaus besitzen diese Objekte Komponenten, die die Darstellungs- und Materialeigenschaften definieren. Beispielsweise verweist eine Render-Komponente auf ein bestimmtes Material oder eine bestimmte Geometrie, die dem Objekt zugeordnet ist, und Lichtquellen enthalten zusätzlich eine Licht-Komponente, die die Intensität und Farbe des Lichts bestimmt.

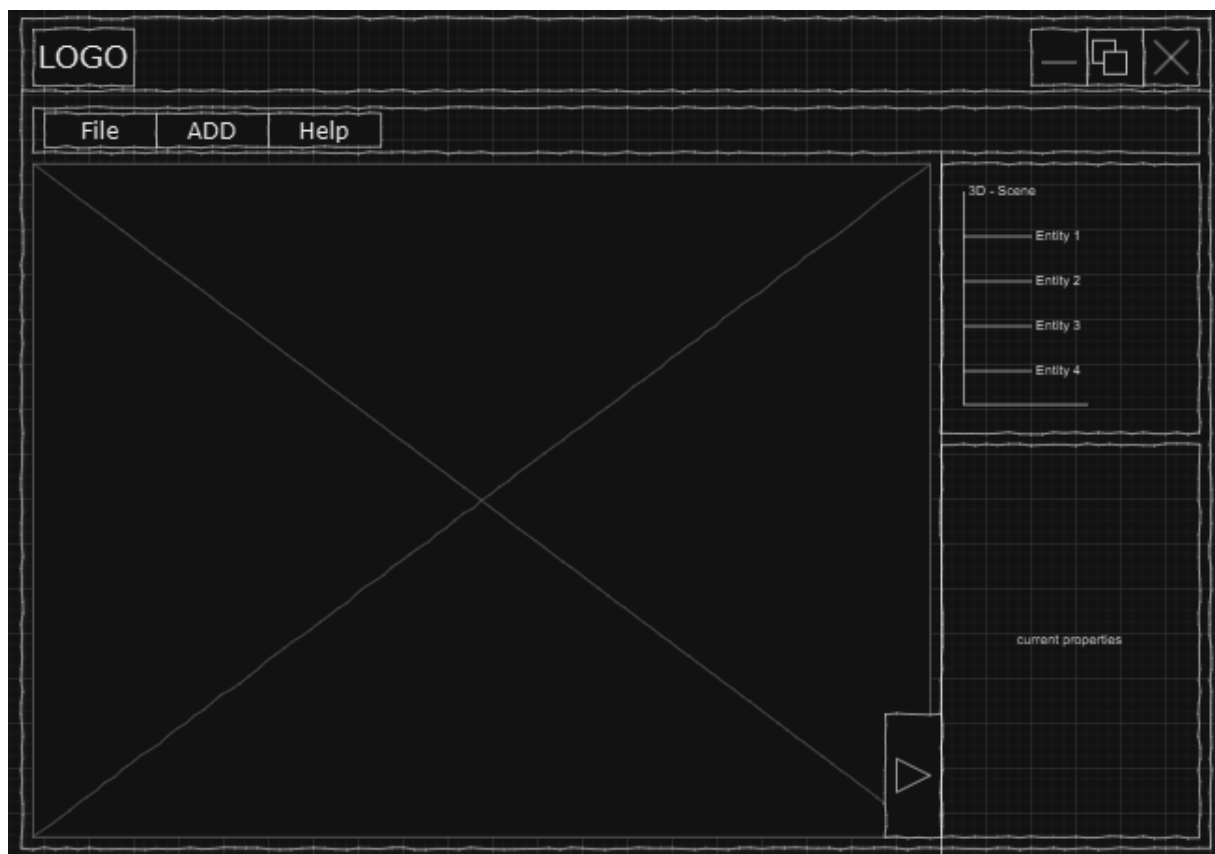
Die Ressourcen verweisen auf interne Dateien mit einem relativen Pfad, die für die Geometrie- und Materialdefinitionen der Objekte in der Szene benötigt werden. Zu den Ressourcen gehören Verweise auf Geometrien wie OBJ-Dateien, die das Modell eines Objekts beschreiben, und auf Materialien, die Texturen oder andere Oberflächeneigenschaften enthalten. Diese Dateien liefern wichtige Informationen über die physikalischen Eigenschaften der Oberflächen, damit der Raytracer das Aussehen der Objekte so realistisch wie möglich berechnen kann.

Zusammengefasst bietet dieses JSON eine strukturierte Datenbasis für die Beschreibung und den Aufbau einer 3D-Szene. Durch die Definition der wichtigsten Eigenschaften der Objekte in der Szene, wie Position, Material und Beleuchtung, ermöglicht es einem Raytracer, diese Informationen zu verwenden, um die Szene korrekt zu beleuchten und zu rendern.

8. Graphische Gestaltung und Nutzungskonzept

Im Folgenden werden einige grafische Gestaltungskonzepte der einzelnen Dialoge durch Mock-Up-Zeichnungen dargestellt. Textuelle Beschreibungen veranschaulichen zudem die Übersicht der Dialoge.

8.1 Hauptbildschirm

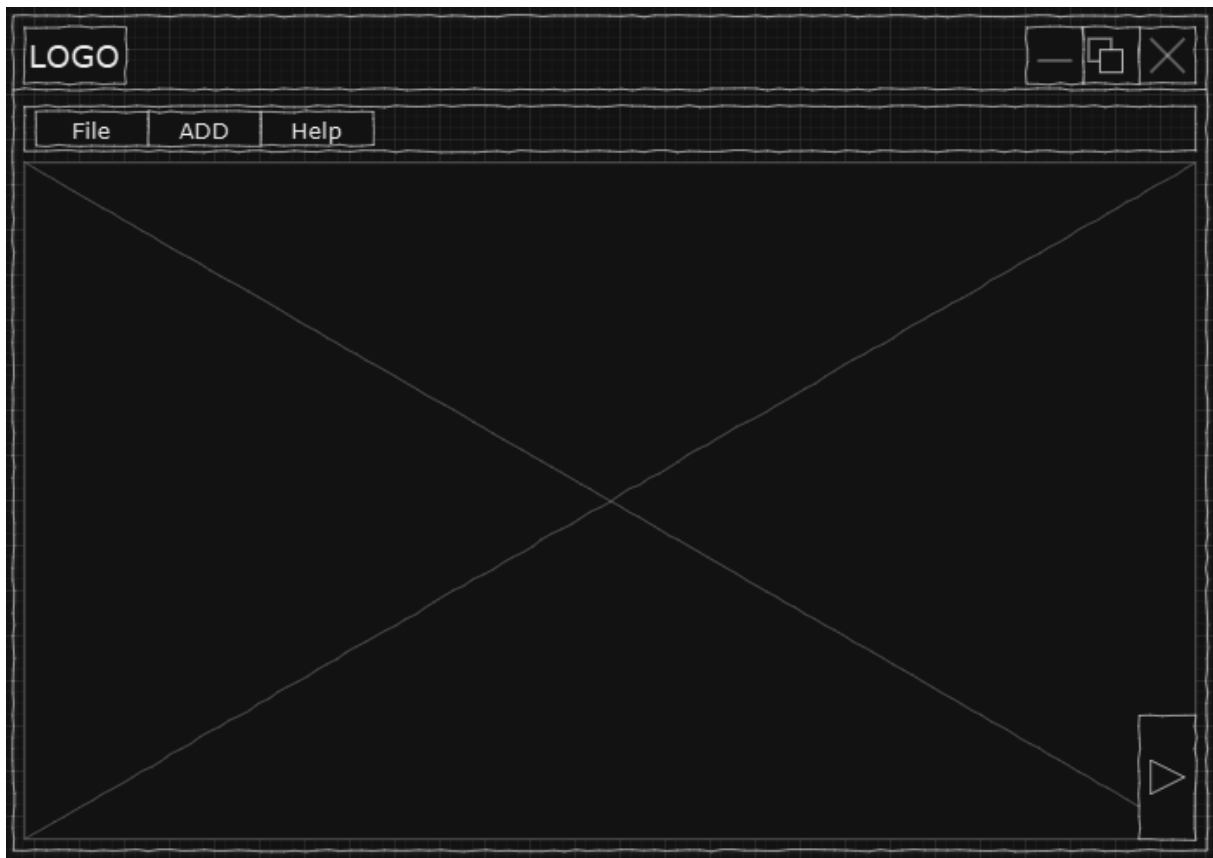


Unser Hauptbildschirm verfügt oben über einen Reiter, über den man auf weitere Informationen, Einstellungen oder funktionale Anforderungen zugreifen kann. Im Menü befinden sich die Einträge „File“ und „Add“, die jeweils als Dropdown-Menüs gestaltet sind und weiter unten detailliert aufgelistet und erklärt werden.

An der rechten Seite finden sich wichtige Informationen zur aktuellen 3D-Szene. Darunter ist ein Entity-Tree platziert, und direkt darunter werden weitere Informationen zur aktuellen 3D-Szene angezeigt, einschließlich Informationen zur eventuell geladenen JSON-3D-Szene. Diese Informationen lassen sich durch einen Button am Rand, der in der Mitte einen Pfeil enthält, einklappen und wieder ausklappen.

Hier sehen Sie eine Mockup-Zeichnung im eingeklappten Informationszustand:

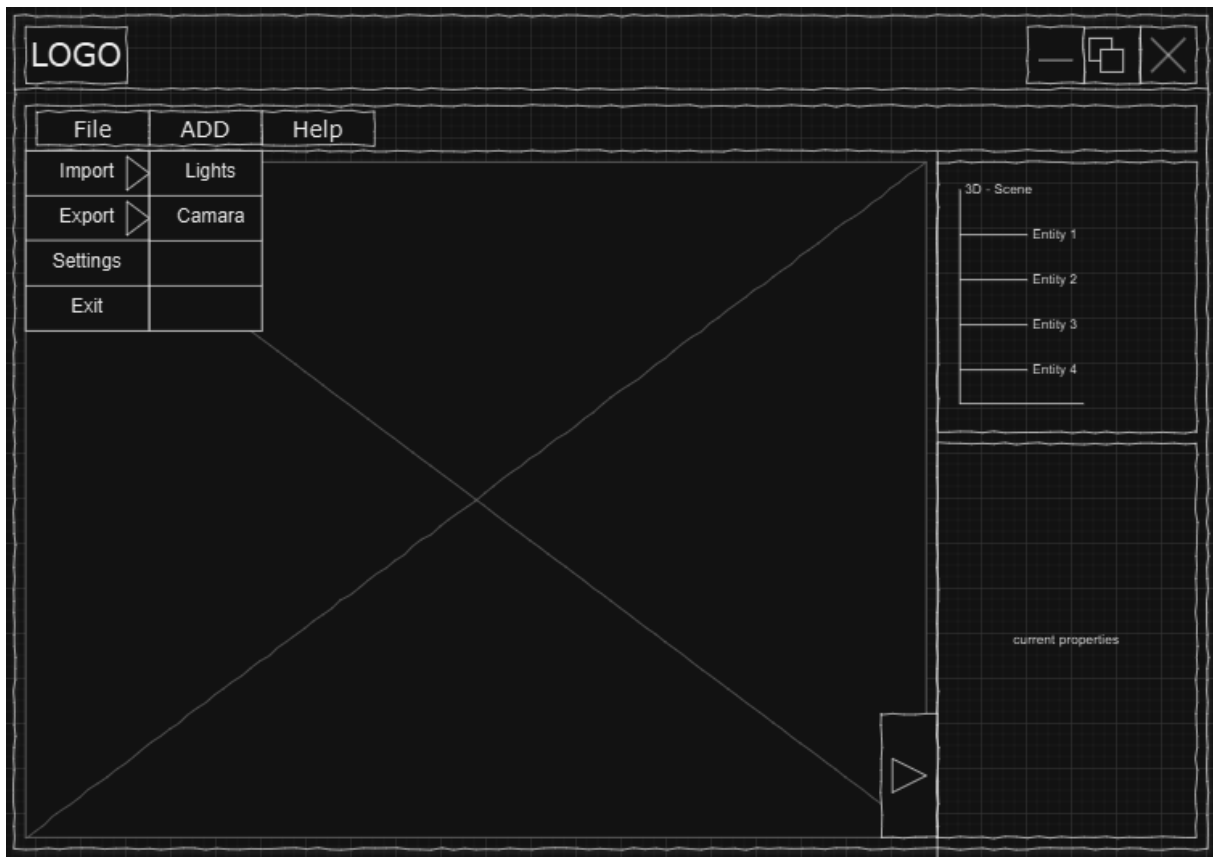
1. Eingeklappten Informationszustand:



Hier sind die Dropdown-Menüs der beiden Reiter im Detail dargestellt: Der Reiter *File* enthält die Optionen Import, Export, Settings und Exit, die den Zugriff auf verschiedene Datei- und Anwendungseinstellungen ermöglichen. Im Reiter *Add* befinden sich die Optionen Lights und Camera, mit denen sich Beleuchtungselemente sowie Kameras in die 3D-Szene hinzufügen lassen.

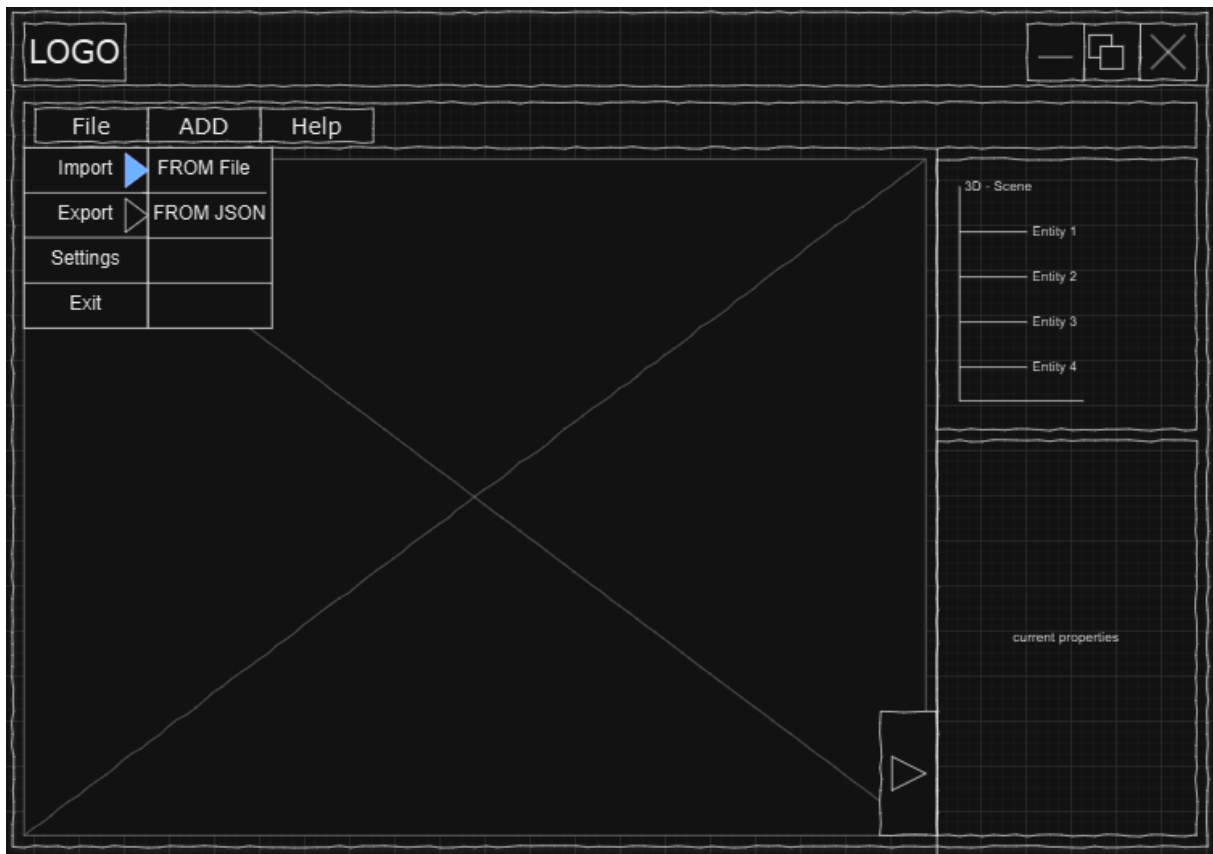
Die Optionen Lights und Camera bieten die Möglichkeit, entweder neue Lichtelemente oder Kameras zu erstellen oder bereits vorhandene Elemente in den Fokus zu nehmen. Sobald ein Element fokussiert ist, kann es mit der Maus bewegt und präzise in der 3D-Szene positioniert werden.

2. Ausgeklappte Reiter „File“ und „Add“:



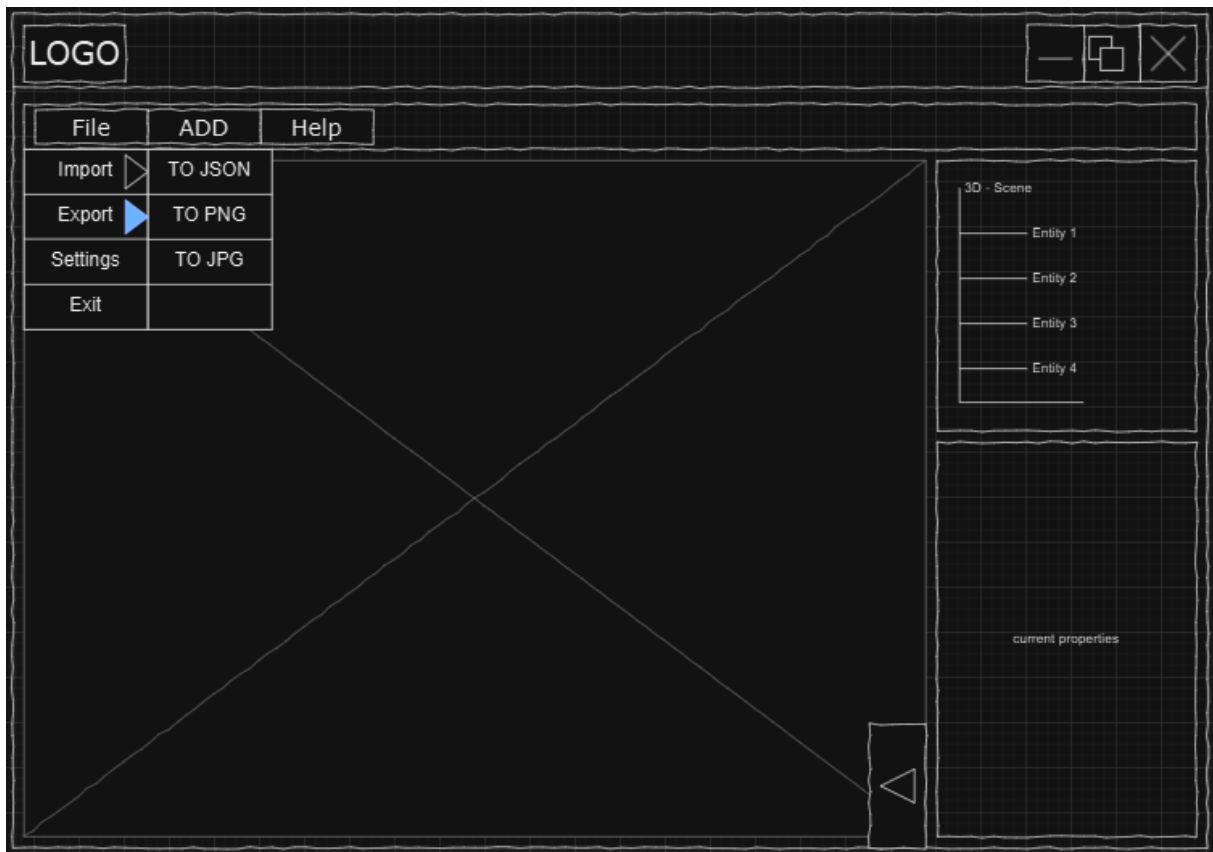
Nun werfen wir einen Blick auf die Optionen, die sich durch Anklicken oder Hover-Effekt beim Menüpunkt „Import“ öffnen. Hier hat der Nutzer die Wahl, entweder eine Datei oder eine 3D-Szene im JSON-Format zu öffnen bzw. zu importieren.

3. Ausgeklappte Import-Option:



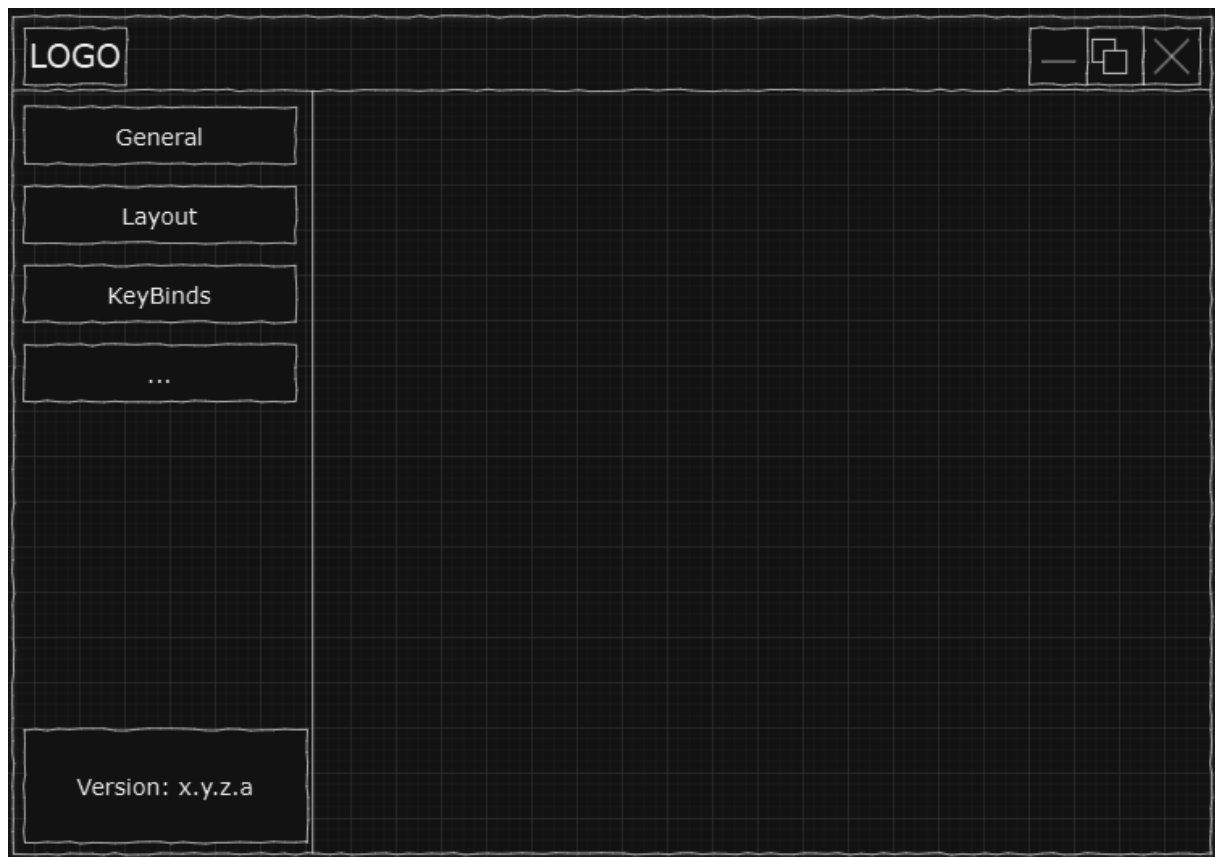
Beim Menüpunkt „Export“ stehen nach dem Anklicken oder Hovern mehrere Exportoptionen zur Verfügung. Der Nutzer kann hier entscheiden, in welchem Format die Daten gespeichert werden sollen. Zur Auswahl stehen die Formate JSON, PNG und JPG, wodurch sowohl strukturierte Daten als auch Bilddateien generiert werden können, je nach gewünschtem Verwendungszweck und Zielplattform. Diese flexible Auswahl ermöglicht es, die exportierten Inhalte an spezifische Anforderungen oder Weiterverarbeitungsprogramme anzupassen.

4. Ausgeklappte Export-Option:



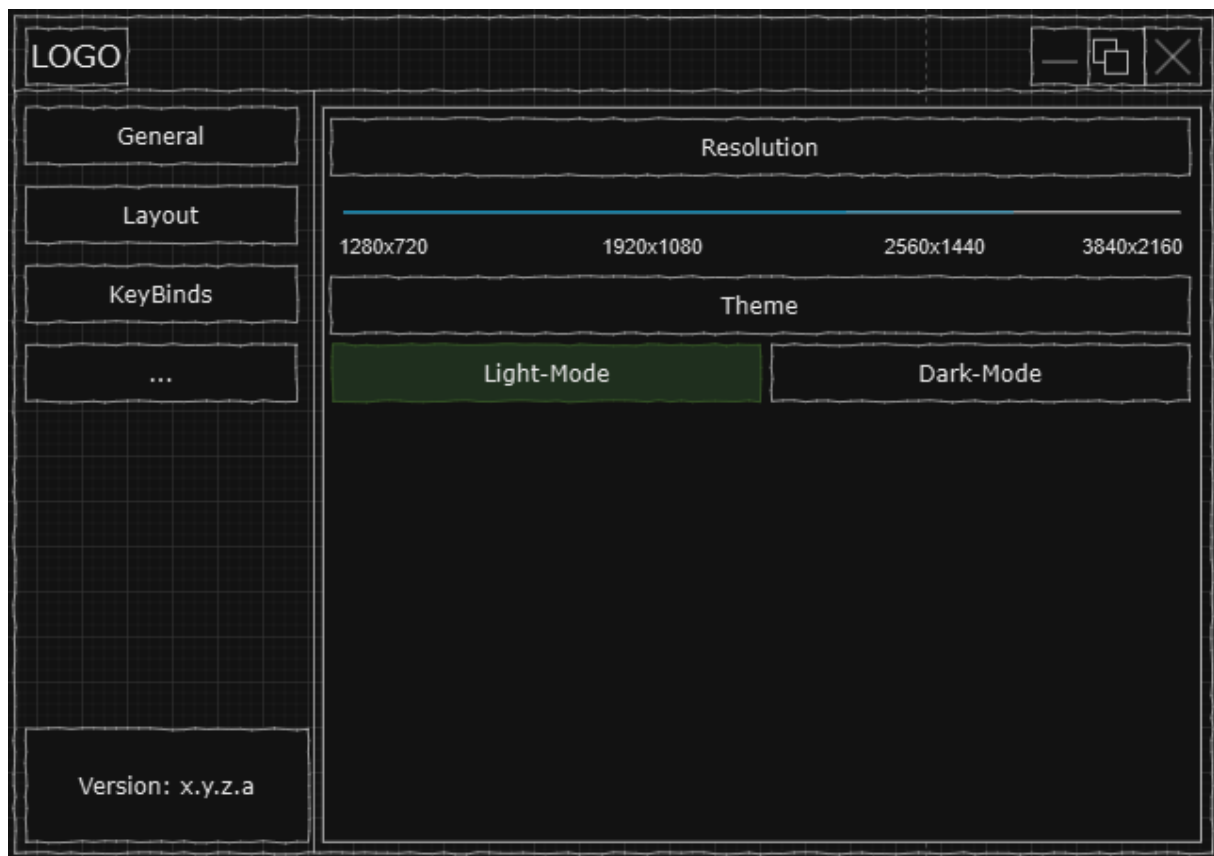
Nun kommen wir zur Option „Settings“. Durch das Öffnen dieser Option wird eine neue Oberfläche angezeigt, die den Hauptbildschirm überlagert und eine eigenständige Struktur aufweist. Diese Struktur enthält verschiedene Einstellungsmöglichkeiten, deren Aufbau und Funktionen später noch im Detail erklärt werden.

8.2 Einstellungen (gestrichen)



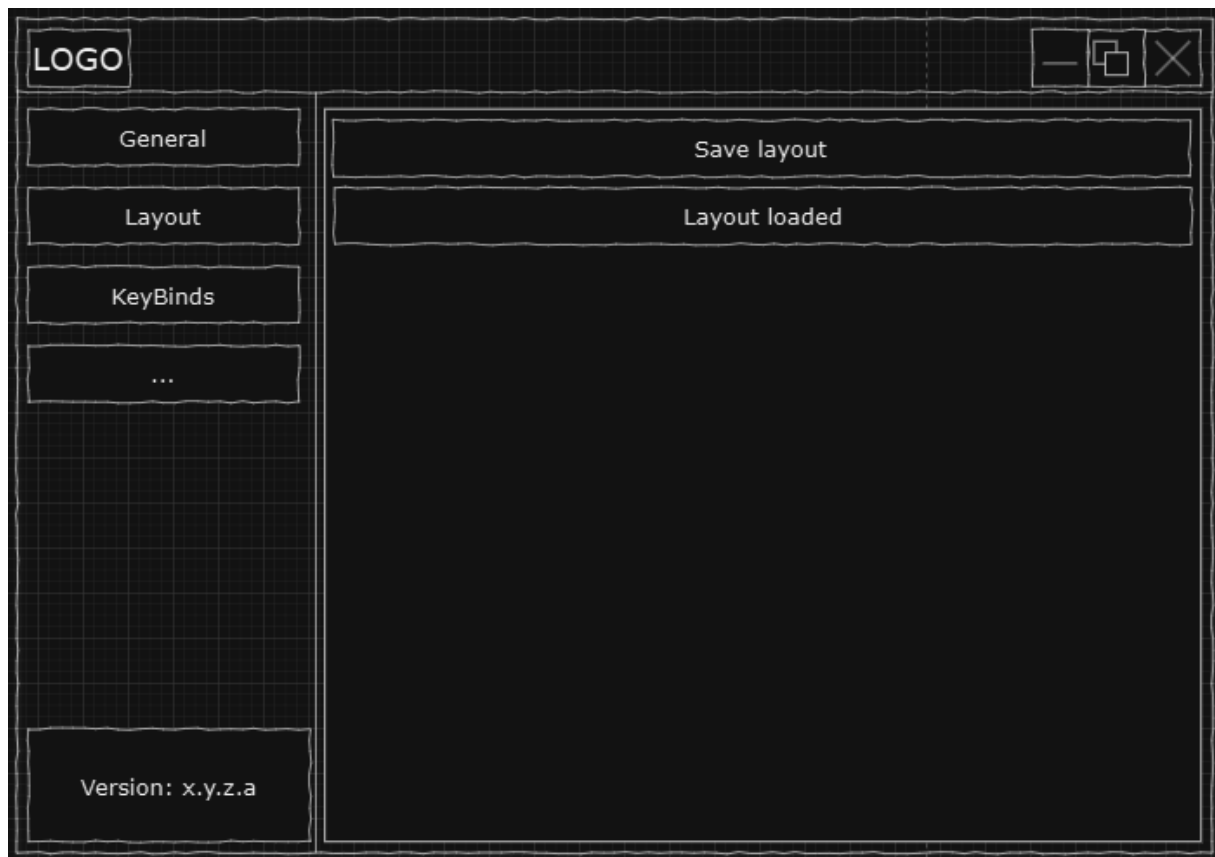
Mein Settings-Fenster weist eine einfache, aber moderne Struktur auf. Auf der linken Seite sind die Optionen wie „General“, „Layout“ und „Keybinds“ angeordnet. Wenn eine dieser Optionen angeklickt wird, werden auf der rechten Seite im großen Kontextfenster alle zugehörigen Eigenschaften und Möglichkeiten angezeigt. Diese übersichtliche Anordnung ermöglicht es den Nutzern, schnell zwischen den verschiedenen Einstellungskategorien zu wechseln und die gewünschten Anpassungen vorzunehmen.

5. General:



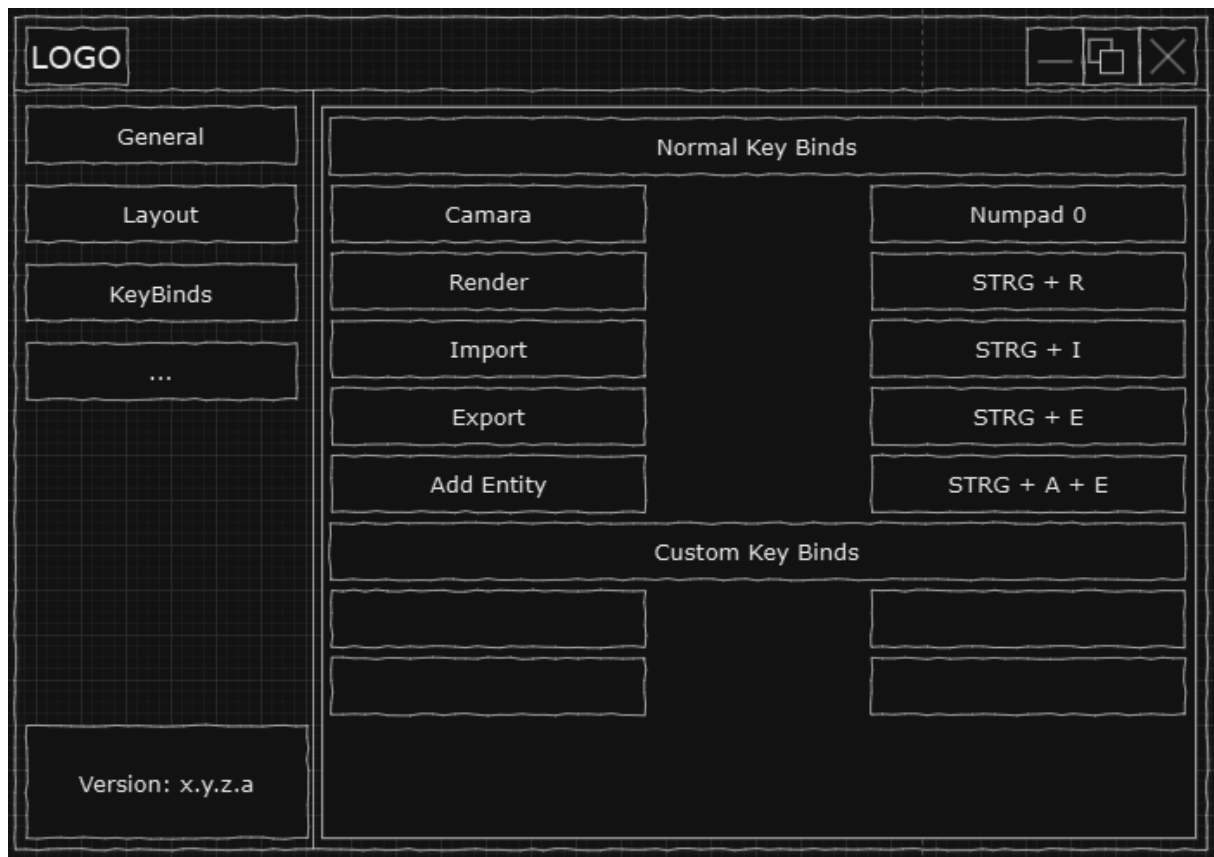
Wenn die Option „General“ ausgewählt wird, erhält der Nutzer die Möglichkeit, die Auflösung bzw. „Resolution“ der Anwendung anzupassen. Außerdem kann gewählt werden, ob die Anwendung im „Light Theme“ oder „Black Theme“ genutzt werden soll. Durch das Anklicken der jeweiligen Theme-Option wird diese hervorgehoben, sodass die aktuelle Einstellung klar angezeigt wird. Diese intuitive Gestaltung erleichtert die Anpassung der Benutzeroberfläche entsprechend den individuellen Vorlieben.

6. Layout:



Wenn die Option „Layout“ ausgewählt wird, hat der Nutzer die Möglichkeit, die Anordnung der NanoGUI-Elemente auf dem Hauptbildschirm zu speichern, da diese frei bewegbar sind. Außerdem steht ein Standard-Layout als Vorschlag zur Verfügung, das zu Beginn geladen wird. Dies ermöglicht es den Nutzern, entweder mit der vorgeschlagenen Anordnung zu arbeiten oder ihre eigene, personalisierte Anordnung zu erstellen und abzuspeichern.

7. Key-Binds:



Bei der Option „Key Binds“ hat der Nutzer die Möglichkeit, bereits vorgeschlagene Tastenkombinationen einzusehen und nach seinen Wünschen zu bearbeiten. Darüber hinaus stehen zusätzliche benutzerdefinierte Slots zur Verfügung, sodass der Nutzer auch eigene Tastenkombinationen festlegen kann, falls er spezielle Anforderungen oder Vorlieben hat. Diese Flexibilität ermöglicht eine individuelle Anpassung der Steuerung und verbessert die Benutzererfahrung erheblich.

9. Sources

9.1 Contents

Auf unserer [GitHub-Seite](#) finden Sie eine Übersicht der Quellen sowie weiterführende Informationen zu unserem Projekt. Dort können Sie den Code einsehen, herunterladen und sich einen detaillierten Einblick in die Implementierung verschaffen.