

IHK Abschlussprüfung Teil 2 – Sommer 2025	 Industrie- und Handelskammer zu Rostock
IT- Berufe Deckblatt für die Dokumentation zur Betrieblichen Projektarbeit	Ausbildungsberuf: Bitte Beruf auswählen:

Titel der Betrieblichen Projektarbeit: Simulationsprojekt: „Murmelsimulation – Wie beeinflussen physikalische Parameter die Murmel?“
--

Prüfungsteilnehmer/in	Ausbildungs-/ Praktikumsunternehmen:
Vor- und Familienname: Benjamin Seidel Anschrift des Prüfungsteilnehmers: OT: Klicken Sie hier, um den Ortsteil einzugeben. Straße: Friedrich-Wolf-Straße 27 PLZ Ort 18435 Telefon: -491729289623 E-Mail: benjamin.159@web.de	Verantwortliche/r für die Durchführung des betrieblichen Auftrages: Klicken Sie hier, um Vor- und Zunamen des Ansprechpartners im Unternehmen einzugeben. Anschrift des Unternehmens: OT: Klicken Sie hier, um den Ortsteil einzugeben. Straße: Klicken Sie hier, um die Straße einzugeben. PLZ Ort Klicken Sie hier, um Postleitzahl und Ort einzugeben. Telefon: Klicken Sie hier, um die Telefonnr. des Ansprechpartners einzugeben. E-Mail: Klicken Sie hier, um die E-Mail-Adresse des Ansprechpartners einzugeben.

_____	Klicken Sie hier, um ein Datum einzugeben. _____	_____
Ort	Datum	Unterschrift Antragsteller/-in (Prüfling)
_____	Klicken Sie hier, um ein Datum einzugeben. _____	_____
Ort	Datum	Stempel/ Unterschrift Verantwortliche/r für den betrieblichen Auftrag

Inhaltsverzeichnis

Abkürzungsverzeichnis.....	1
1. Einleitung.....	2
1.1. Ausgangssituation und Problemstellung.....	2
1.2. Projektziele.....	2
1.3. Rahmenbedingungen und Abgrenzungen.....	2
1.4. Anforderungen an die Anwendung.....	3
1.4.1 Funktionale Anforderungen.....	3
1.4.2 Nicht-funktionale Anforderungen.....	4
2. Projektplanung.....	5
2.1. Vorgehensmodell.....	5
2.2. Ressourcen- und Ablaufplanung.....	5
2.2.1. Zeitplanung.....	5
2.2.2. Kostenplanung.....	6
2.3. Risikoanalyse.....	6
3. Technische Umsetzung.....	7
3.1. Architektur und Design.....	7
3.1.1. Architektur-Konzept (Hybrid / Actor-Component).....	7
3.1.2. Datenauswertung.....	8
3.2. Auswahl der Technologien.....	8
3.2.1. Engine.....	8
3.2.2. Integrierte Entwicklungsumgebung.....	8
3.2.3. Modellierungssoftware.....	9
3.3. Implementierung.....	10
3.3.1. Kernkomponenten.....	10
3.3.2. Zufallselemente und Verteilung.....	10
3.4. Benutzeroberfläche.....	10
3.4.1. UI-Konzept und Benutzbarkeit.....	10
3.4.2. Elemente und Animation.....	10
4. Qualitätssicherung.....	11
4.1. Testkonzept.....	11
4.2. Testdurchführung.....	11
4.3. Validierung der Zufallsmodelle.....	11
4.4. Soll-Ist-Vergleich.....	11
5. Zusammenfassung und Ausblick.....	12
5.1. Fazit.....	12
5.2. Ausblick.....	12

Literaturverzeichnis.....	13
Anhang.....	i
Tabellen:.....	i



Abkürzungsverzeichnis



1. Einleitung

1.1. Ausgangssituation und Problemstellung

Das hier dokumentierte Projekt „Gravity Rollers“ ist die vorgegebene Projektarbeit für das dritte Ausbildungsjahr.

Das Kernziel der Anwendung ist die Entwicklung einer physikbasierten, zeitabhängigen Simulation. Dem Anwender soll ein Werkzeug bereitgestellt werden, um die Auswirkungen von konfigurierbaren physikalischen Parametern und definierten Zufallsfaktoren auf einer Murmelbahn visuell zu analysieren und auszuwerten.

Die Realisierung erfolgt als Windows-Anwendung mit grafischer Benutzeroberfläche. Eine wesentliche technische Herausforderung ergibt sich aus der Anforderung, die Lauffähigkeit auf den Schulrechnern als Zielsystem zu gewährleisten.

1.2. Projektziele

Die primären Projektziele umfassen die Entwicklung einer intuitiven und benutzerfreundlichen grafischen Benutzeroberfläche.

Ebenso zentral ist die nachvollziehbare visuelle Darstellung der Simulation. Der Anwender muss die Auswirkungen der individuell konfigurierten physikalischen Parameter sowie der implementierten Zufallsereignisse auf das Endergebnis analysieren und auswerten können.

Die Abgabe der Anwendung erfolgt als einzelne, auf Windows lauffähige, ausführbare Datei.

1.3. Rahmenbedingungen und Abgrenzungen

Das Projekt unterliegt zeitlichen, technischen und organisatorischen Rahmenbedingungen, die vorgegeben sind.

Zeitlicher und Organisatorischer Rahmen: Das Projekt ist im Rahmen der schulischen Projektarbeit durchzuführen. Der fixe Abgabetermin für die Dokumentation und das fertige Softwareprodukt ist der 06.02.2026.

Technische Rahmenbedingungen: Als Zielplattform ist eine ausführbare 64-Bit-Windows-Anwendung (.exe) definiert. Eine zwingende Anforderung ist die performante Lauffähigkeit der Simulation auf der Standard-Hardware der Schulrechner.

Inhaltliche Abgrenzung: Um den Projektumfang im vorgegebenen Zeitrahmen realisieren zu können, werden folgende Funktionalitäten explizit ausgeschlossen:

- Es wird kein Speichersystem für rennübergreifende Statistiken implementiert.



- Ein Level-Editor zur Erstellung eigener Strecken ist nicht Bestandteil des Projekts.
- Die Anwendung wird als reine Single-User-Simulation konzipiert.
- Der Fokus liegt auf der Konfiguration und Auswertung eines einzelnen Rennlaufs.

1.4. Anforderungen an die Anwendung

1.4.1 Funktionale Anforderungen

Die funktionalen Anforderungen definieren den konkreten Leistungsumfang der Anwendung. Sie beschreiben die spezifischen Aufgaben und Funktionen, die das System erfüllen muss.

- **Zeitabhängige Simulation:** Die Kernfunktionalität muss eine Physik-Simulation sein, deren Ablauf und Ergebnis von der verstrichenen Zeit abhängen.
- **Konfigurierbarkeit:** Dem Anwender müssen mindestens sieben physikalische Eingabeparameter zur Verfügung stehen..
- **Zufallsfaktoren:** Es sind mindestens drei unterschiedliche Zufallsverteilungen zu implementieren, um unvorhersehbare Einflüsse auf die Simulation abzubilden.
- **Geschwindigkeitssteuerung:** Die Abspielgeschwindigkeit der Simulation muss durch den Benutzer in mindestens drei Stufen regelbar sein.
- **Visualisierung & Auswertung:** Der Simulationsverlauf ist grafisch darzustellen. Nach Abschluss eines Durchlaufs müssen die Ergebnisse visuell aufbereitet werden.
- **Animation:** Es ist mindestens eine thematisch passende Animation zu integrieren.
- **Bereitstellung:** Die Anwendung muss als eigenständige, ausführbare Datei kompiliert werden, die auf der definierten Zielhardware lauffähig ist.

1.4.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen legen die Qualitätskriterien fest, unter denen die Anwendung betrieben und weiterentwickelt wird. Sie definieren die Rahmenbedingungen für die technische Umsetzung, die Qualität der Benutzerinteraktion sowie die Wartbarkeit.

- **Benutzbarkeit:** Die Gestaltung der grafischen Benutzeroberfläche und die Interaktionsführung müssen die Prinzipien der ISO-Norm 9241-110 berücksichtigen.
- **Code-Qualität:** Der Quellcode ist nach den Kriterien von „Clean Code“ zu strukturieren, um Lesbarkeit, Wartbarkeit und Erweiterbarkeit sicherzustellen.
- **User Experience:** Durch den gezielten Einsatz von Feedback-Elementen soll eine positive Nutzererfahrung gewährleistet werden.



2. Projektplanung

2.1. Vorgehensmodell

Die Realisierung des Projekts erfolgt anhand des Wasserfallmodells. Diese Entscheidung für ein lineares, sequenzielles Vorgehen ist durch die Rahmenbedingungen begründet: Der Abgabetermin ist fix auf den 06.02.2026 terminiert und der Funktionsumfang wurde im Projektauftrag als unveränderliche Anforderung (Fixed Scope) definiert.

Das Modell sieht vor, dass die Entwicklung in aufeinanderfolgenden, logisch abgegrenzten Phasen erfolgt, wobei jede Phase erst nach Abschluss der vorhergehenden begonnen wird:

1. **Analysephase:** Detaillierung der Anforderungen.
2. **Entwurfsphase:** Planung der Architektur und des UI-Designs.
3. **Implementierungsphase:** Programmierung der Simulationslogik sowie der GUI.
4. **Qualitätssicherung:** Durchführung von Funktionaltests und Soll-Ist-Vergleich.
5. **Dokumentation:** Erstellung der Projektdokumentation.

Durch diese strikte Phasenorientierung wird eine exakte Überwachung des Projektfortschritts gewährleistet (siehe Zeitplanung in Kap. 2.2.1). Ein entscheidender Vorteil dieses Ansatzes liegt in der Prävention von Scope Creep (ungeplante Umfangserweiterung), da die Anforderungen bereits vor Beginn der Implementierung finalisiert werden. Agile Methoden wie Scrum wurden bewusst verworfen, da sie ihre Stärken primär in iterativen Feedback-Schleifen ausspielen. Das inhärente Risiko des Wasserfallmodells, die späte Identifikation von Fehlern, wird durch eine intensivierte Analyse- und Designphase proaktiv minimiert.

2.2. Ressourcen- und Ablaufplanung

2.2.1. Zeitplanung

Für das Projekt steht ein Gesamtzeitraum von 23 Wochen (06.09.2025 bis 06.02.2026) zur Verfügung. Um die Vergleichbarkeit mit betrieblichen Projekten zu gewährleisten, wird ein Netto-Arbeitsaufwand von 160 Stunden veranschlagt. Dies entspricht einer durchschnittlichen Arbeitszeit von 7 Stunden pro Woche.

Die Phasen verteilen sich wie folgt auf den Projektzeitraum:

Verdeutlicht in Schaubild 1: Gantt-Diagramm zur Zeitplanung



2.2.2. Kostenplanung

Die Kostenplanung betrachtet die theoretischen Entwicklungskosten des Projekts. Da es sich um ein internes Ausbildungsprojekt handelt, werden keine Gewinne oder Umsatzsteuern ausgewiesen. In der Kostenplanung werden die Personalkosten, anhand des Stundesatzes und die Sachmittelkosten berücksichtigt.

Gesamtkosten:

Veranschaulicht in Tabelle 1: Kostenplanung

2.3. Risikoanalyse

Zur Sicherstellung des Projekterfolgs wurde im Vorfeld eine detaillierte Betrachtung möglicher Störfaktoren durchgeführt. Ziel dieses Risikomanagements ist es, technische und organisatorische Unsicherheiten frühzeitig zu erkennen und durch definierte Gegenmaßnahmen handhabbar zu machen. Die Bewertung der einzelnen Risiken erfolgt dabei anhand ihrer Eintrittswahrscheinlichkeit sowie der potenziellen Auswirkung auf die Projektziele.

Zusammengetragen in Tabelle 2: Risikoauflistung



3. Technische Umsetzung

3.1. Architektur und Design

3.1.1. Architektur-Konzept (Hybrid / Actor-Component)

Das Projekt folgt dem Actor-Component-Modell der Unreal Engine, erweitert um eine hybride Schichtenarchitektur aus C++ und Blueprints. Dieses Entwurfsmuster wurde gewählt, um eine klare Trennung der Verantwortlichkeiten zu gewährleisten und gleichzeitig die Performanz der Simulation mit der Flexibilität der visuellen Gestaltung zu vereinen.

Die Architektur gliedert sich in drei logisch getrennte Bereiche:

- **Logik-Ebene:** Es kapselt die gesamte Simulationslogik, die physikalischen Berechnungen und die Zustandsdaten. Durch die Implementierung in C++ wird die notwendige Rechenleistung für die Physik-Engine sichergestellt.
- **Visuelle Ebene:** Die Darstellung und das Feedback an den Nutzer erfolgen über abgeleitete Blueprint-Klassen und das UMG-Framework.
- **Steuerungs-Ebene:** Der APlayerController dient als Schnittstelle zwischen Eingabegerät und Logik.

Betrachtung von Alternativen: Als Alternative wurde eine reine Blueprint-Architektur (Visual Scripting) in Betracht gezogen. Diese hätte die Einstiegshürde gesenkt und eine schnellere Prototypen-Entwicklung ermöglicht. Allerdings wurde dieser Ansatz verworfen, da komplexe mathematische Berechnungen in Blueprints zu einer schlechteren Wartbarkeit und signifikanten Performance-Einbußen führen würden. Gerade im Hinblick auf die begrenzten Ressourcen der Schulrechner bietet der gewählte Hybrid-Ansatz den entscheidenden Vorteil. Zudem erleichtert die Kapselung der Physik in C++ das spätere Debugging und Testen der Kernfunktionen.

3.1.2. Datenauswertung

3.2. Auswahl der Technologien

3.2.1. Engine

Da die realistische Darstellung physikalischer Kräfte die Kernanforderung des Projekts darstellt, lag der Fokus der Bewertung auf der Simulationsgenauigkeit.

Kriterien für die Nutzwertanalyse:



- **Physik-Engine (Gewichtung: 4):** Für eine physikalische Simulation ist die Qualität und Stabilität der Physik-Engine das wichtigste Kriterium.
- **GUI-Framework (Gewichtung: 3):** Die Anforderung einer ISO-konformen Benutzeroberfläche erfordert leistungsfähige, integrierte UI-Werkzeuge.
- **Programmiersprache (Gewichtung: 2):** Die Unterstützung einer performanten Sprache ist wichtig für die Optimierung.
- **Dokumentation & Community (Gewichtung: 2):** Wichtig für die Problemlösung.
- **Einarbeitungszeit (Gewichtung: 1):** Da die technische Eignung im Vordergrund steht, wird der Einarbeitungsaufwand als weniger kritisch eingestuft.

Durchführung in Tabelle 3: Nutzwertanalyse Engine.

Begründung der Entscheidung: Die Wahl fiel auf die Unreal Engine 5. Ausschlaggebend war der Hybrid-Ansatz. Unity und Godot boten insbesondere bei der integrierten Physik-Performance Nachteile im Vergleich zur Chaos Engine.

3.2.2. Integrierte Entwicklungsumgebung

Da das Projekt auf einem C++ Hybrid-Ansatz basiert, ist eine leistungsfähige IDE für das Schreiben, Refactoring und Debuggen des Codes essenziell.

Kriterien für die Nutzwertanalyse:

- **Unreal-Integration (Gewichtung: 4):** Die IDE muss das komplexe Unreal-Build-System und spezifische C++ Makros verstehen.
- **Performance (Gewichtung: 3):** Schnelles Indizieren und ein zuverlässiger Debugger sind für flüssiges Arbeiten nötig.
- **Kosten (Gewichtung: 2):** Die Wirtschaftlichkeit muss betrachtet werden.
- **Plugins (Gewichtung: 1):** Die Verfügbarkeit von nützlichen Zusatztools.

Durchführung in Tabelle 4: Nutzwertanalyse IDE.

Begründung der Entscheidung: Trotz der grundsätzlichen Kostenpflicht fiel die Wahl auf JetBrains Rider. Der entscheidende technische Vorteil ist die tiefe Unreal-Integration. Der wirtschaftliche Nachteil entfällt durch eine kostenlose Bildungslizenz. Visual Studio Code wurde aufgrund der aufwendigen manuellen Konfiguration verworfen.

3.2.3. Modellierungssoftware

Für die Erstellung und Bearbeitung der 3D-Assets (Murmeln, Streckenteile) war die Wahl einer geeigneten 3D-Software notwendig.

Kriterien für die Nutzwertanalyse:



- **Kompatibilität (Gewichtung: 4):** Der Export zur gewählten Engine muss reibungslos funktionieren.
- **Lizenzkosten (Gewichtung: 4):** Da für diesen Bereich kein Budget vorgesehen ist, sind kostenlose oder Open-Source-Lösungen stark zu bevorzugen.
- **Funktionsumfang (Gewichtung: 2):** Grundlegende Funktionen müssen vorhanden sein.

Durchführung in Tabelle 5: Nutzwertanalyse Modellierungssoftware .

Begründung der Entscheidung: Die Entscheidung fiel auf Blender. Neben dem ausreichenden Funktionsumfang war die Lizenzfreiheit das entscheidende Argument. Zudem bietet Blender inzwischen eine exzellente Integration in den Unreal-Workflow, was Konvertierungsprobleme minimiert.

3.3. Implementierung

3.3.1. Kernkomponenten

3.3.2. Zufallselemente und Verteilung

3.4. Benutzeroberfläche

3.4.1. UI-Konzept und Benutzbarkeit

3.4.2. Elemente und Animation



4. Qualitätssicherung

4.1. Testkonzept

4.2. Testdurchführung

4.3. Validierung der Zufallsmodelle

4.4. Soll-Ist-Vergleich



5. Zusammenfassung und Ausblick

5.1. Fazit

5.2. Ausblick



Literaturverzeichnis

Anhang

Tabellen:

Position	Berechnung	Betrag
Personalkosten	160 Std. × 5,00 €	800,00 €
Software-Lizenzen	Open-Source: Blender, UE5 Bildungslizenz: Rider	0,00 €
Hardware-Nutzung	Bestand	0,00 €
Gesamtsumme		800,00 €

Tabelle 1: Kostenplanung

Risiko	Eintrittswahrscheinlichkeit	Auswirkung	Maßnahmen
Performance auf Zielhardware	Hoch	Hoch	Kein Lumen/Nanite, Static Lighting, Physik in C++
Fehler beim Packaging (Build)	Mittel	Hoch	Frühzeitige Test-Builds, Regelmäßige Log-Prüfung
Technische Komplexität	Mittel	Mittel	Offizielle Doku nutzen, Blueprint-Fallback
Kollisionsfehler (Tunneling)	Mittel	Hoch	CCD aktivieren, Sub-Stepping erhöhen
Modellierungsfehler (Physik)	Mittel	Mittel	Plausibilitätschecks, Manueller Soll-Ist-Vergleich
Falsche Zufallsverteilung	Niedrig	Mittel	Testreihen visualisieren, Histogramm-Validierung
Scope Creep	Hoch	Niedrig	Einhaltung "Fixed Scope", Priorisierung

Tabelle 2: Risikoauflistung

Kriterium	Gew.	Unreal Engine	Unity	Godot	Begründung
Physik-Engine	4	3 (Chaos Physics)	2 (PhysX)	1 (Basis)	Unreal bietet mit Chaos Physics die robusteste integrierte Lösung.
GUI-Framework	3	3 (UMG)	2 (UI Toolkit)	2 (Control Nodes)	Das Unreal Motion Graphics Framework bietet einen visuellen Designer.
Programmiersprache	2	3 (C++ / BP)	1 (C#)	2 (GDScript)	Der Hybrid-Ansatz vereint maximale Performance mit schneller Iteration.
Dokumentation & Community	2	3 (Sehr gut)	3 (Sehr gut)	2 (Gut)	Exzellente offizielle Dokumentation und riesige Community.
Einarbeitungszeit	1	3 (Vorwissen)	2 (Basis)	1 (Keine)	Vorkenntnisse sind vorhanden.
Gesamtnutzwert		36	26	19	

Tabelle 3: Nutzwertanalyse Engine

Kriterium	Gew.	JetBrains Rider	Visual Studio	VS Code	Begründung
UE-Integration	4	3 (Tief)	2 (Gut)	1 (Basis)	Zeigt Blueprint-Bindung direkt im Code.
Performance	3	3 (Schnell)	2 (Mittel)	3 (Schnell)	Deutlich schnellere Indizierung als Visual Studio.
Kosten	2	1 (Teuer*)	3 (Kostenlos)	3 (Kostenlos)	*Nachteil wird durch kostenlose Bildungslizenz ausgeglichen.
Plugins	1	2 (Spezifisch)	2 (Viel)	3 (Riesig)	Rider benötigt kaum Plugins, da UE-Tools integriert sind.
Gesamtnutzwert		25	23	19	

Tabelle 4: Nutzwertanalyse IDE

Kriterium	Gew.	Blender	Maya	Cinema 4D	Begründung
Kompatibilität	4	3 (Sehr gut)	3 (Standard)	2 (Gut)	Blender bietet eine direkte Schnittstelle.
Lizenzkosten	4	3 (Kostenlos)	1 (Teuer)	1 (Teuer)	Blender ist Open Source.
Funktionsumfang	2	3 (Umfangreich)	3 (Profi)	3 (Profi)	Alle Tools bieten mehr als für das Projekt benötigt wird.
Gesamtnutzwert		30	22	18	

Tabelle 5: Nutzwertanalyse Modellierungssoftware

Bilder:

Startdatum: 06.09.2025 | Enddatum: 06.02.2026
Dauer: 23 Wochen | Geplante Stunden: 160h

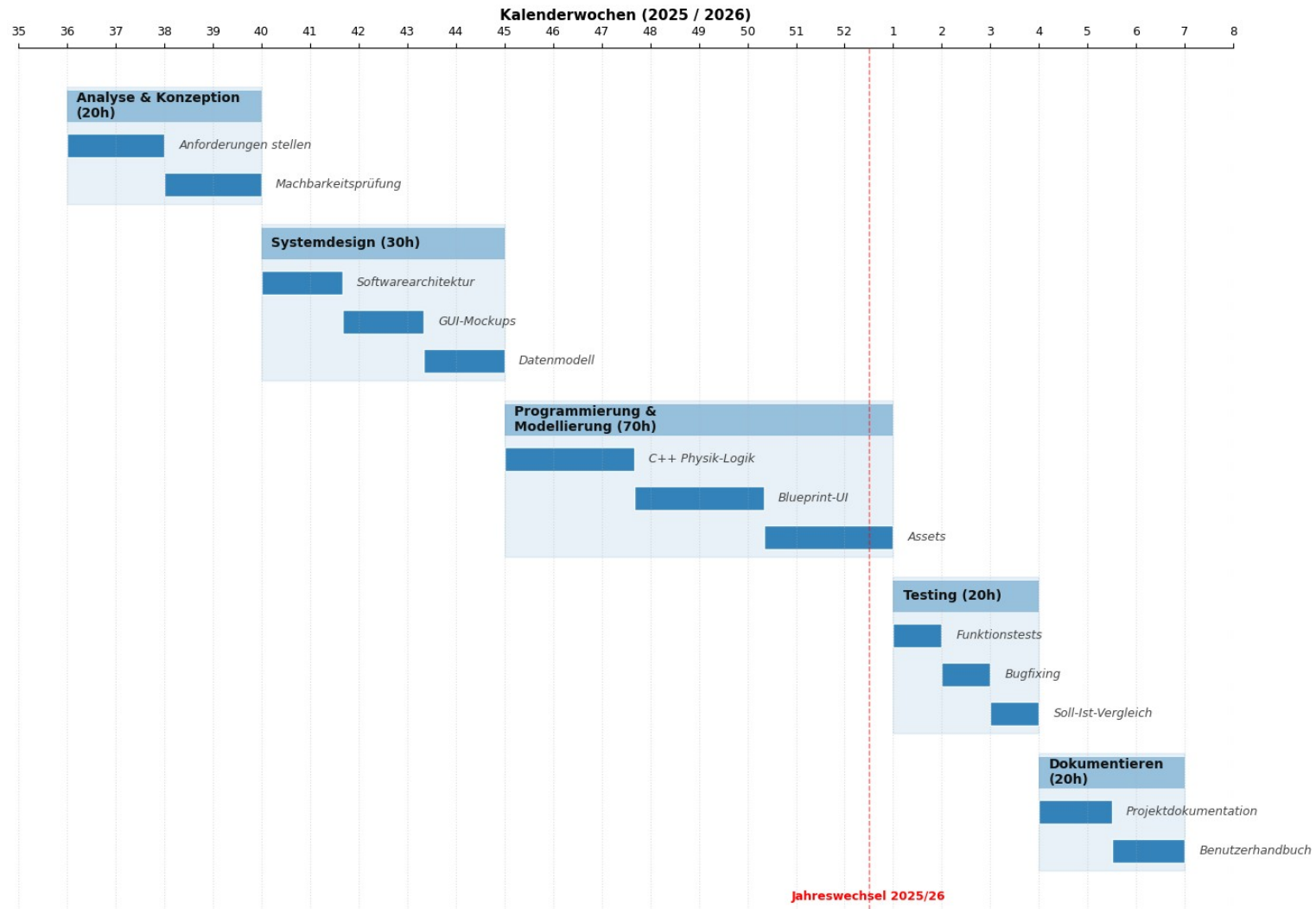


Schaubild 1: Gantt-Diagramm zur Zeitplanung