Mad Machines – Anforderungsdokument

Sophia Backert und Niklas Fengler
SE-Praktikum 2025

Mai 2025

Spielübersicht

Mad Machines ist ein physikbasiertes 2D-Puzzlespiel, bei dem die Spielenden Maschinenkomponenten – wie Zahnräder, Rampen, Förderbänder und Bälle – kombinieren, um bestimmte Aufgaben zu lösen. Dabei muss mit Hilfe dieser Komponenten ein Zielzustand erreicht werden. Das Spiel fördert Experimentierfreude, logisches Denken und ein Verständnis für einfache Mechanik. Ziel dieses Projekts ist die Konzeption und Umsetzung eines reduzierten, aber erweiterbaren Physikpuzzlespiels, das sich an der Idee von *Crazy Machines* orientiert. Die Anwendung soll durch eine modulare Architektur eine spätere Integration KI-basierter Interaktionen ermöglichen. Fokus wir dabei auf die Physiksimulation, eine intuitive Benutzeroberfläche sowie umfassende Funktionen zur Erstellung eigener Level gelegt.

Benutzerinteraktion

Die Spielenden interagieren über zwei Hauptmodi mit dem Spiel:

- **Editiermodus:** Komponenten können frei platziert und rotiert werden.
- **Puzzlemodus:** Vorgegebene Level mit begrenzten Bauteilen und Platzierungsbeschränkungen müssen zum Erreichen des Zielzustandes geführt werden.

Alle Interaktionen erfolgen visuell und mausgesteuert (Drag & Drop) über eine intuitive grafische Benutzeroberfläche mit direktem Feedback.

Pflichtfeatures

Physiksimulation und Editiermodus: Der zentrale Spielablauf wechselt zwischen dem Platzieren von Objekten und dem Klicken auf “Play”, um die Simulation zu starten.

- Statische und dynamische Physikobjekte sowie Spezialobjekte (siehe Kick-off Folien)
- Button zum Starten/Abbrechen der Simulation
- Zurücksetzen in den Editiermodus
- Editiermodus mit simplen Platzieren und Entfernen von Objekten

Levelsystem: Levels sollen per Datei modifizierbar und teilbar sein.

- Unterstützung für Level-Dateien
- Dateien sollten menschenlesbar sein (z.,B. JSON)
- Level-Progression nach Abschließen eines Levels im Puzzlemodus

Einfache Grafik: "Programmierer-Grafik" ist ausreichend.

- Objekte als einfache geometrische Formen darstellen
- Farben zur besseren Unterscheidbarkeit hinzufügen
- Anzeige in Echtzeit

Puzzlemodus: Einrichtung eines Spielmodus mit folgenden Elementen:

- Siegesbedingungen (z.,B. Ball erreicht Zielbereich)
- Einschränkungszonen (z.,B. Platzierungsverbote)
- Inventarsystem mit begrenzter Anzahl an Objekten

Wahlfeatures

Wahlfeatures dienen der Vertiefung. Drei davon müssen umgesetzt werden. Wir entscheiden unns für richtige Menüs, dem verbesserten Level-Editor und ein Maschienen-Interface.

I. Richtige Menüs

- Hauptmenü als zentrale Anlaufstelle
- Levelauswahlmenü mit Thumbnails und Levelinfos

II. Mehr Objekte

- Seil, statischer Anker, Hebel
- Erweiterung von Ballon und Eimer um Befestigungspunkte

III. Maschinen-Interface

- Zustandsprotokoll (Text/Console)
- Headless Mode / Kommandozeilen-Interface

IV. Level-Editor

- Drag&Drop
- Platzierte Objekte verschieben und rotieren
- Undo-/Redo-Funktion

V. Komplexere Grafiken

- Objekte mit Texturen oder Bildern darstellen
- Möglichkeit zum Austausch von Sprite-Sheets/Texture-Packs
- Hintergrundgrafiken einfügen

Nicht-Funktionale Anforderungen

Codequalität

- Einhaltung definierter Metriken gemäß den Vorgaben im Abschnitt *Coding Style, Metriken, Testabdeckung.*
- Konsistenter Stil gemäß der vereinbarten Java-Style-Convention.
- Ausreichende Kommentierung des Codes an nicht-trivialen Stellen.

Testabdeckung

- Ziel: Mindestens 80% Testabdeckung (gemessen via JaCoCo), ausgenommen GUI-spezifischer Code.
- Testabdeckung erfolgt durch Unit-Tests mit JUnit 5.

Dokumentation

- Laufend gepflegte Anforderungsdokumentation (Story Cards, Anwendungsfalldiagramme).
- Javadoc-Dokumentation aller öffentlichen Klassen und Methoden.
- Technische Architektur-Dokumentation mit UML-Diagrammen (z. B. Klassen-, Sequenz-, Zustandsdiagramme), wo sinnvoll.
- Benutzerhandbuch zur Bedienung der Anwendung (ggf. ergänzt durch Screenshots und Beispiele).