Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Pflichtenheft und technische Spezifikation im Programmierprojekt

Sokoban Reloaded

Mitarbeiter: Eray Aslan

Heltonn Ngalemotchaleu

Maximilian Cerwinski

Robert Braatz

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Inhaltsverzeichnis

1 Visionen und Ziele

Um einen ersten Überblick über die Use-Cases der Software zu schaffen, wurde eine Anforderungsliste erstellt und mit dem Auftraggeber abgestimmt. Die Liste wurde hierbei in die Punkte „Sokoban-Spiel“, „Level-Editor“ & „Allgemein“ unterteilt.

Die in Tabelle 1 abgebildete Anforderungsliste liefert somit eine Grundlage für die Erzeugung des Use-Case Diagramms.

Nr.	Anforderung	Ausprägung	Herkunft
1	Allgemein		Moodle
1.1	Lauffähig auf Windows	hohe Priorität	Moodle
1.2	Lauffähig ohne Zusatzinstallationen oder Administratorenrechte	hohe Priorität	Moodle
1.3	Verwendung von Visual Studio	hohe Priorität	Moodle
2	Level-Editor zur Erstellung eigener Levels		Moodle
2.1	GUI mit Auswahlmöglichkeit Spiel/Editor	hohe Priorität	Implizit
2.2	Objektpositionen speicherbar	hohe Priorität	Implizit
2.3	Lösbarkeit automatisiert prüfen	niedrige Priorität	2. Meeting
2.4	Level Gültigkeit checken (Anfangsbedingungen & Finalbedingungen)	hohe Priorität	2. Meeting
2.5	Toolbox mit prinziprelevanten Objekten	hohe Priorität	Implizit
3	Spiel Sokoban		Moodle
3.1	4 Default-Level mit steigendem Anspruch	hohe Priorität	1. Meeting
3.2	Fortschritt soll im Level speicherbar sein	hohe Priorität	1. Meeting
3.3	Steuerungsvorgabe/Wahl	hohe Priorität	2. Meeting
3.4	Spielfigur animiert	niedrige Priorität	1. Meeting
3.5	Level-Design/Präsentation	niedrige Priorität	2. Meeting
3.6	Automatisierte Testung	mittlere Priorität	2. Meeting
3.7	Statistiken für Spieler (Zeit/Anzahl Schritte)	mittlere Priorität	2. Meeting

Tabelle 1: Anforderungsliste

Ziel des Projekts ist die Entwicklung eines Sokoban-Spiels, das neben dem Spielen ebenfalls das Erstellen eigener Level ermöglicht. Als zusätzlicher Motivationsfaktor sollen Spielerstatistiken und Bestenlisten implementiert werden.

Als technische Grundlage nutzt das Projektteam die bewährte Spiel-Engine Unity, da diese viele benötigte Funktionen bereithält und somit eine planbare und sichere Entwicklung begünstigt.

2 Projektstruktur

2.1 Use-Cases

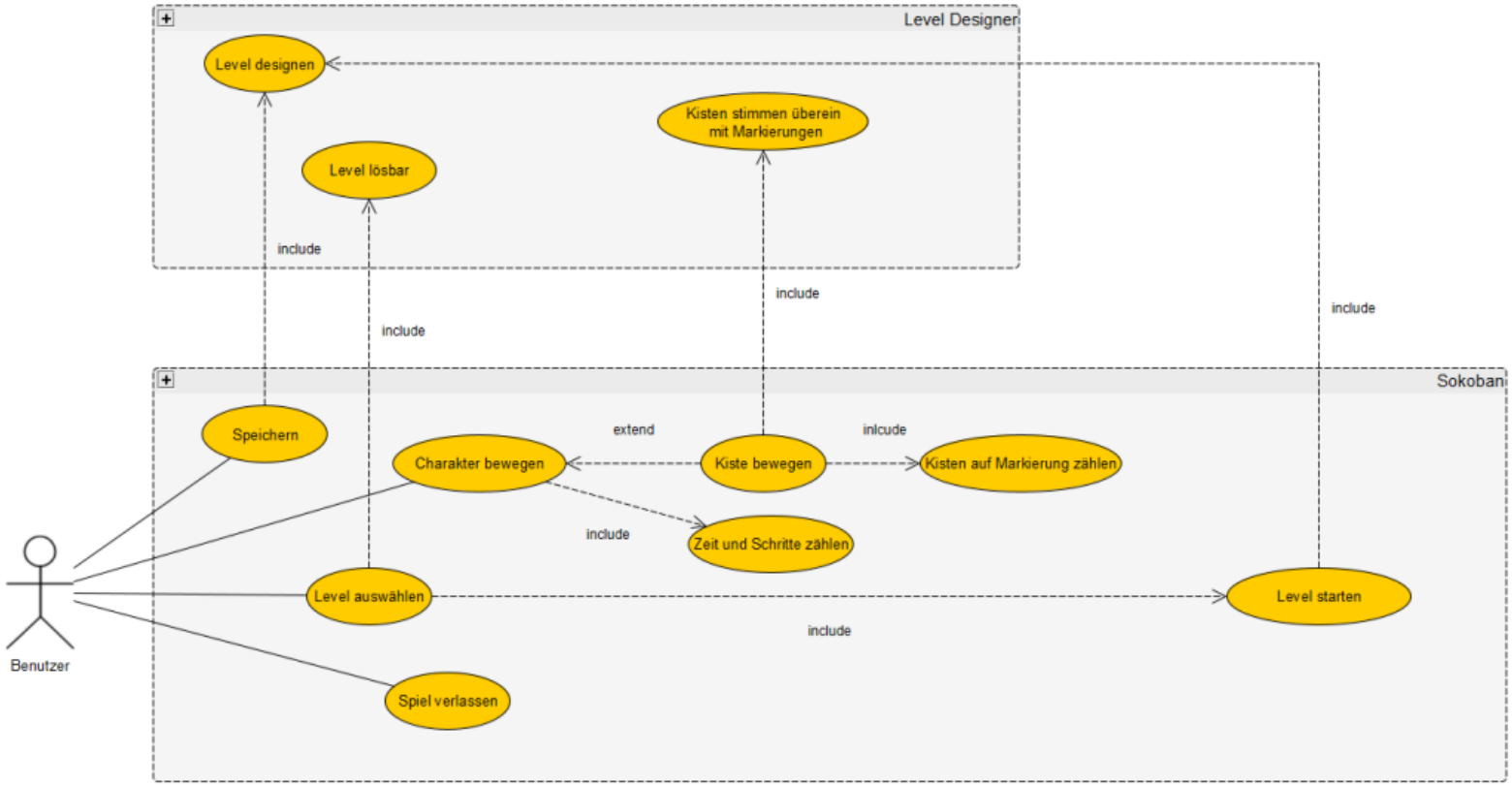


Abbildung 1: Use-Case Diagramm

Das in Abbildung 1 dargestellte Use-Case Diagramm stellt die aus der Anforderungsanalyse abgeleiteten Use-Cases und ihre Beziehungen dar. Ziel dieser Betrachtung ist die Einteilung von Zuständigkeiten. Über diese Zuweisung soll sichergestellt werden, dass kein Use-Case des Programms bei der Abarbeitung der unter Punkt 5 genannten Arbeitsschritte unerfüllt bleibt. *[Siehe dazu Projekt-Gantt-Diagramm]* Der zuständige Projektmitarbeiter prüft somit regelmäßig, ob die Use-Cases seines Arbeitspakets im Laufe der Entwicklung erfüllt werden.

Die Use-Cases lassen sich in den Level-Designer, sowie das eigentliche Sokoban-Spiel unterteilen. Zur Strukturierung der Zuständigkeiten werden mehrere Use-Cases einem übergeordnetem *Arbeitspaket* zugeordnet. Der Projektfortschritt soll im Projektverlauf durch arbeitspaketübergreifende Meilensteine regelmäßig synchronisiert werden. Die Festlegung der Meilensteine erfolgt unter Punkt 5.

2.2 Zuteilung von Arbeitspaketen

Arbeitspaket 1 – Nutzeroberfläche fasst hierbei folgende Use-Cases zusammen:

1A	Level auswählen
1B	Speichern
1C	Spiel verlassen
1D	Nächstes Level starten

Für die Erstellung von eigenen Levels werden folgende Use-Cases im *Arbeitspaket 2 – Editor* bearbeitet:

2A	Level Designen
2B	Level Speichern
2C	Level lösbar

Die Use-Cases des Spiels werden im Arbeitspaket 3 – Spiel umgesetzt:

3A	Charakter bewegen
3B	Kiste bewegen
3C	Kisten stimmen überein mit Markierungen
3D	Zeit und Schritte zählen
3E	Kisten auf Markierung zählen

Zusätzlich zu den speziellen Arbeitspaketen ist die Koordination sowie die Schnittstellen-prüfung & -entwicklung im generalistischen *Arbeitspaket 4 – Schnittstellen* zu realisieren.

Da die Use-Case Komplexität je Arbeitspaket variiert, werden die beiden Arbeitspakete „Nutzeroberfläche“ und „Sokoban-Spiel“ von einer Person durchgesetzt. Das Arbeitspaket Koordination und Schnittstellen hingegen vereint die komplexeste Use-Case Kombination, weswegen hier zwei Personen eingeplant werden. Die Aufteilung lässt sich folgender Tabelle entnehmen:

Arbeitspaket	**Projektmitarbeiter**
Nutzeroberfläche	Heltonn Ngalemotchaleu
Level Editor	Maximilian Cerwinski
Sokoban-Spiel	Heltonn Ngalemotchaleu
Koordination & & Schnittstellen	Robert Braatz & Eray Aslan

2.3 Risiken

Eines der Hauptrisiken im Projekt stellt das Erlernen und Nutzen von Unity dar, da es eine neue Engine bereitstellt, mit der das Projektteam keine Erfahrungen vorweisen kann. Um zukünftigen Komplikationen vorzubeugen, wurde der Entschluss getroffen, einen Udemy Crashkurs zu absolvieren, um möglichst schnell eine Basis zu schaffen, auf welcher anschließend die Arbeit in Unity aufgenommen werden kann.

Ein weiteres Risiko besteht darin, dass zu viele Features in das Projekt miteingebunden werden, was dazu führen kann, dass Zeitdruck entsteht oder eventuell der Aufwand für Arbeitspakete überschätzt wird, wodurch es durchaus dazu kommen könnte, dass einige Features nicht erfolgreich abgeschlossen werden.

Um diesem Problem zu entgegnen, wird unter Punkt 5 ein genauer Zeitablaufplan erstellt, um einen effizienten Arbeitsfortschritt zu garantieren. Zudem stellt die erfolgte Aufgabenaufteilung sicher, dass der Aufwand je Mitarbeiter, sowie die Zuständigkeiten klar definiert sind. Hierdurch können Flaschenhälse und langsamer Fortschritt aufgrund Unzuständigkeiten minimiert werden.

Des Weiteren könnten Schwierigkeiten durch Missverständnisse in der Gruppe auftreten. Um dies zu vermeiden, hat das Projektteam neben der Zeitplanung ebenfalls einen Kommunikationsplan erstellt. Dieser sieht ein zweiwöchiges Meeting vor, um sich zu koordinieren und Fragen zu stellen, sowie 9 große Meilensteinmeetings. Hier kann der allgemeine Projektfortschritt von allen Mitarbeitern bewertet werden. Die Erkenntnisse dieser Meilensteinmeetings werden bei der Feinplanung der darauffolgenden Arbeitsetappe berücksichtigt, wodurch ein agiles Projektmanagement ermöglicht wird.

Letztlich könnten Probleme dadurch entstehen, dass die Kundenwünsche falsch interpretiert werden. Um jenes Problem möglichst zu vermeiden, wurden alle erfolgten Kundengespräche dokumentiert und in der Anforderungsliste zusammengefasst. Die genaue Zuweisung von Arbeitspaketen sichert zusätzlich, dass der Blick für die Kundenanforderungen nicht verloren geht.

Ergänzend wurde der Kunde gezielt nach einer Prioritätseinschätzung gebeten, die bei der Projektplanung unter Punkt 5 einen Fokus auf die hoch priorisierten Kundenwünsche ermöglicht, bevor weniger wichtige Funktionen implementiert werden.

2.4 GUI

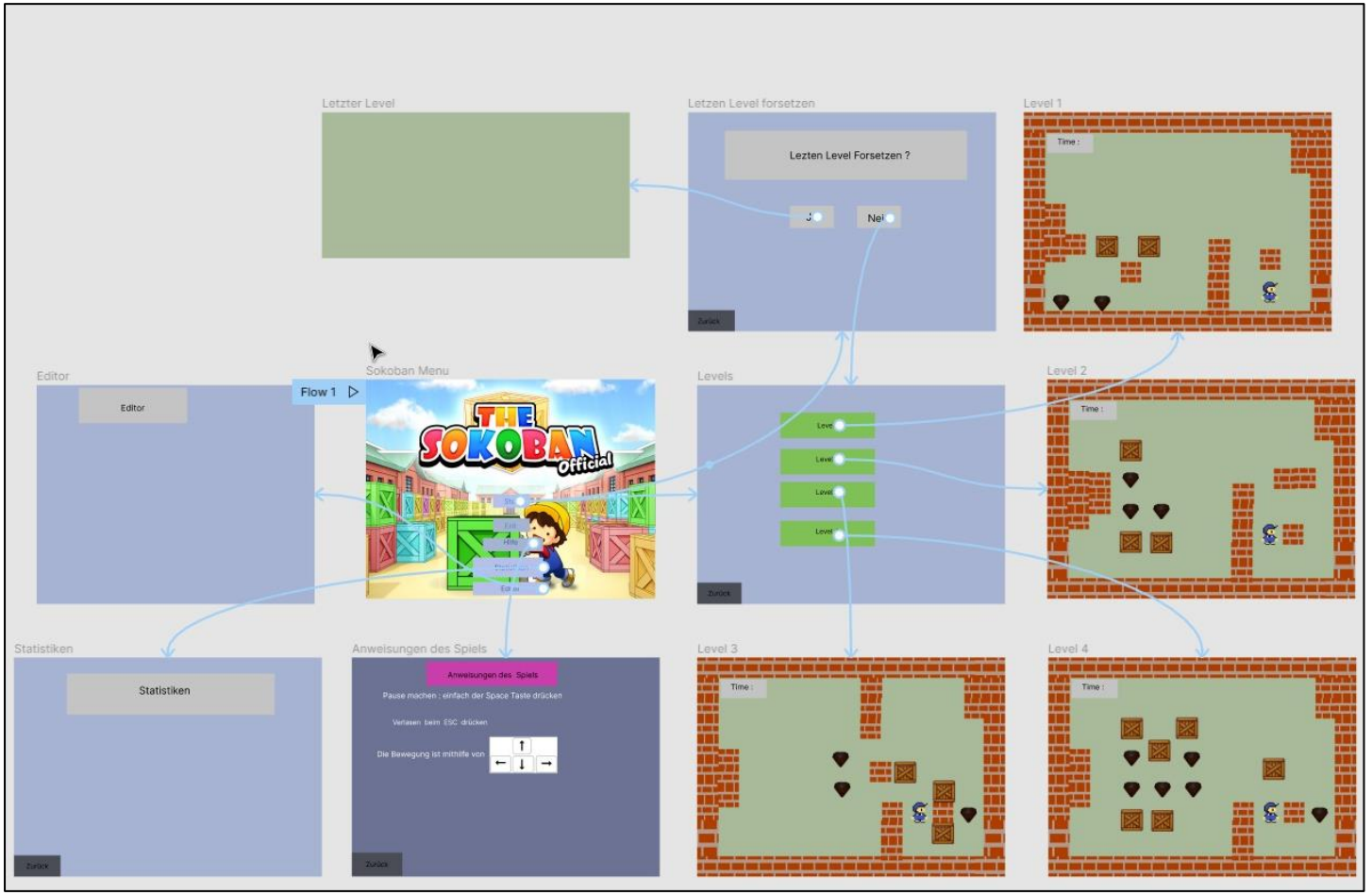


Abbildung 2: Überblick Menüführung

Abbildung 3: Startbildschirm

Der Startbildschirm gibt dem Nutzer Zugriff auf die verschiedenen Funktionen des Spiels. Über die jeweiligen Buttons kann hier das eigentliche Spiel, oder der Editor gestartet werden. Über den Button Hilfe können Hinweise zur Steuerung des Spiels gegeben werden. Der Button Statistiken soll dem Nutzer das Einsehen verschiedener Statistiken gewähren.

Beim Klicken auf Start wird dem Nutzer die Frage gestellt, ob er den zuletzt gewählten Level fortsetzen, oder mit einem neuen beginnen möchte.

Wählt er die Antwort Nein, oder findet das Programm keine Speicherdatei von einer früheren Session, so öffnet sich eine Liste mit verfügbaren Levels. Durch Klicken auf den gewünschten Listeneintrag gelangt man in die Spiel-Ansicht.

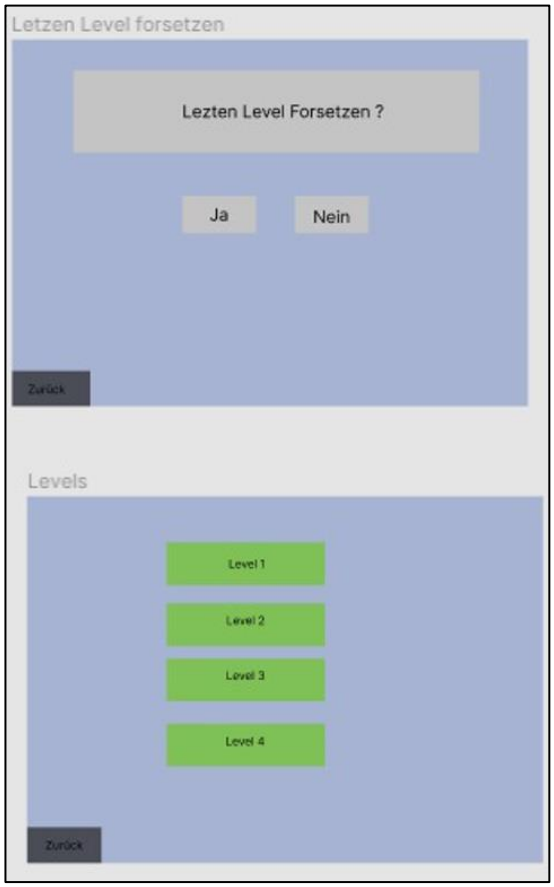
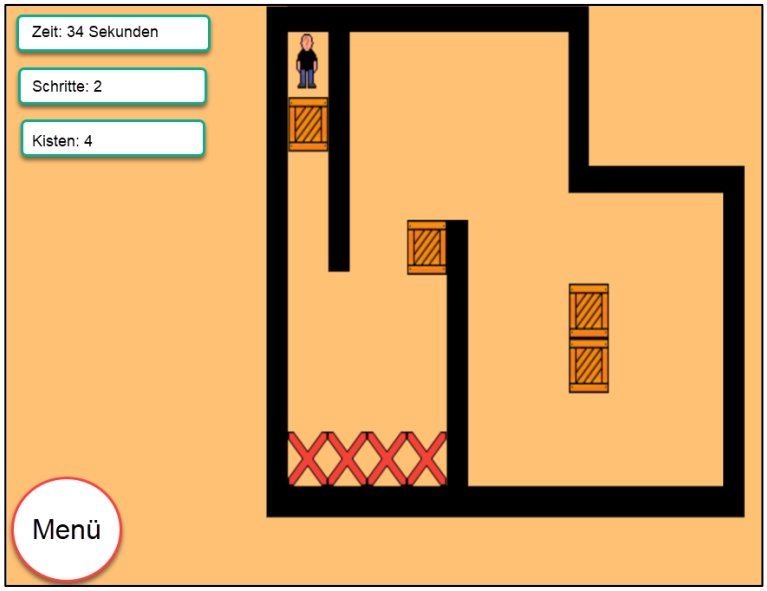


Abbildung 4: Levelabfrage

Die Nutzeroberfläche im eigentlichen Spiel zeigt lediglich die verstrichene Zeit, die gegangenen Schritte, sowie die Anzahl der noch zu positionierenden Kisten. Über den Button „Menü“ lässt sich das Ingame-Menü öffnen.



Das Ingame-Menü ermöglicht dem Spieler den Zugriff auf verschiedene Punkte.

Hier kann das Spiel abgebrochen, oder gespeichert werden. Zudem können Bestenlisten für den jeweiligen Level eingesehen werden.

Abbildung 5: Sokoban-Nutzeroberfläche

Die Nutzeroberfläche des Editors stellt Buttons zur Verfügung, die das Platzieren gewünschter Objekte im Bearbeitungsbereich ermöglicht.

Der Bearbeitungsbereich wird hierbei mit einer Signalfarbe dargestellt. Ist das Level lösbar und endlich, kann das Level über einen zusätzlichen Button gespeichert werden, der ebenfalls das Benennen des jeweiligen Levels ermöglicht.

Die Elemente lassen sich mithilfe der Pfeiltasten auf der Tastatur platzieren. Beim Platzieren werden zudem die Mengen der noch platzierbaren Objekte je Typ berechnet und dem Spieler angezeigt.

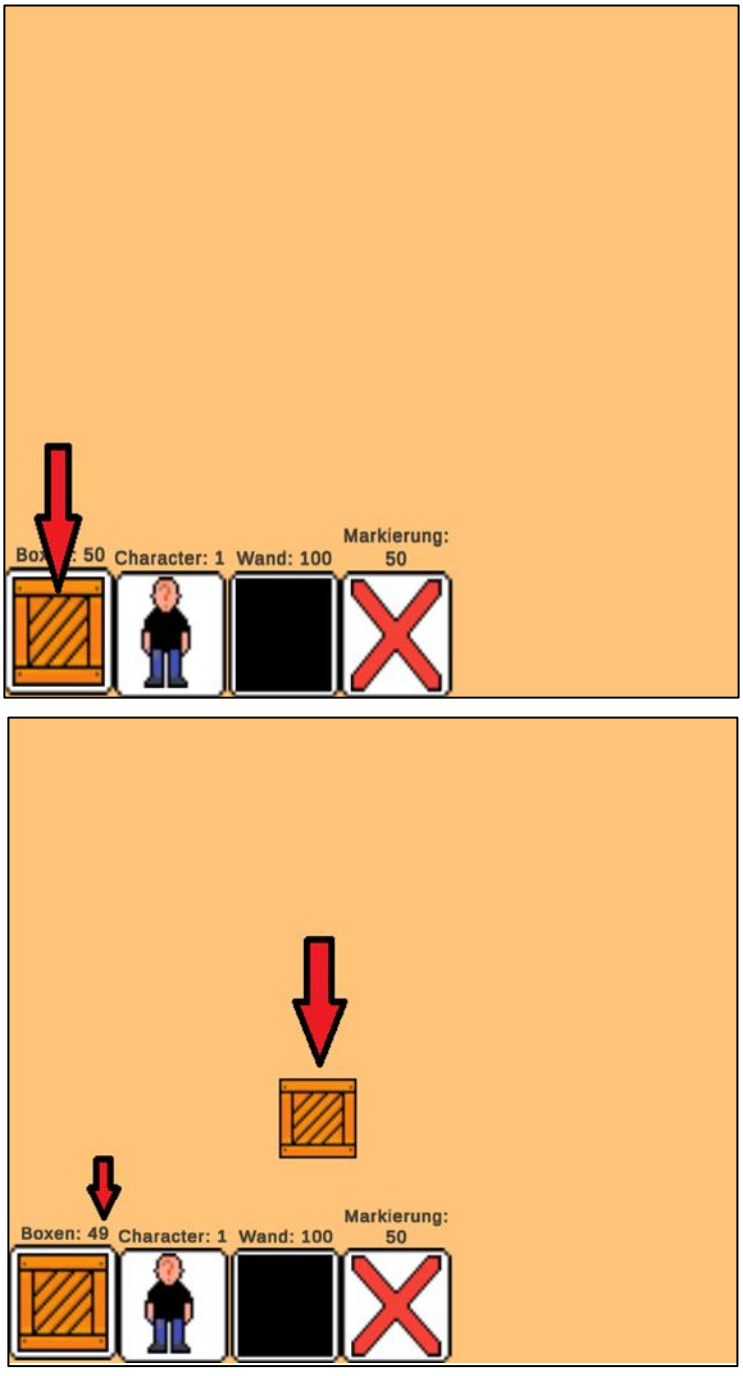


Abbildung 6: Editor-Nutzeroberfläche

3 Realisierung

3.1 Komponentendiagramm

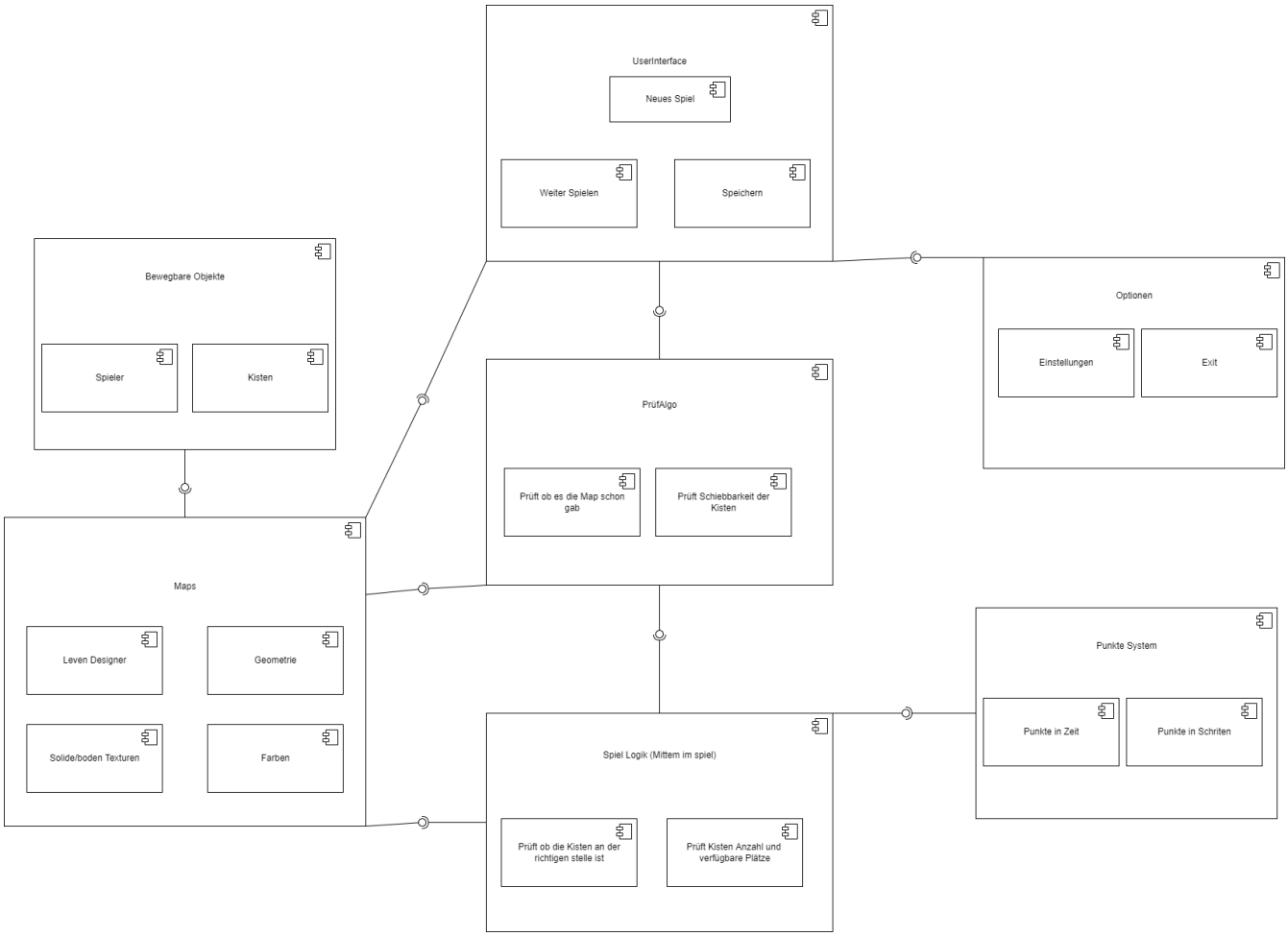


Abbildung 7: Komponenten-Diagramm

Beim Start des Spieles werden zunächst die [Optionen] aufgerufen, dazu gehören [Einstellungen], [Exit], [Neues Spiel], [Speichern], [Weiter Spielen]. [Einstellungen] ermöglicht eine nutzerspezifische Konfiguration des Anwendungsverhaltens. [Exit] beendet die Anwendung.

[Neues Spiel] ruft Informationen von [Bewegbare Objekte], [Maps] und [PrüfAlgo] ab, um der Spiellogik die benötigten Ressourcen zum Starten zur Verfügung zu stellen . Zunächst wird die Spielfläche generiert mit [Maps], dann der Spieler und die Kisten mit [Bewegbare Objekte]. Im Anschluss prüft [PrüfAlgo], ob die Zielbedingungen erfüllt sind und der Level somit erfolgreich abgeschlossen wird.

Beim [Speichern] werden die aktuellen Positionen der [Bewegbare Objekt] gesichert und der Aufbau der Karte [Maps] gesichert um die Funktion [Weiter Spielen] aufrufen zu können. Beim [Weiter Spielen] werden die Informationen abgerufen, die durch [Speichern] persistent abgelegt wurden, um den alten Spielstand weiter spielen zu können. Am Ende jedes Leves gibt es eine Punktebewertung die berechnet wird mit [Punkte System]. Die Punkte nach Schritten [Punkte in Schritten] und nach Zeit [Punkte in Zeit] werden addiert und dann angezeigt.

3.2 Interne Schnittstellen

Die internen Schnittstellen lassen sich gemäß Komponentendiagramm und dazugehörigen Erläuterungen zu folgenden Abhängigkeiten im Sinne von Ausgabe-Komponente übergibt Information an Eingabe-Komponente zusammenfassen:

Bewegbare Objekte	----->	UserInterface
Optionen	----->	UserInterface
Maps	----->	UserInterface
Bewegbare Objekte	----->	Maps
UserInterface	----->	PrüfAlgo
Maps	----->	PrüfAlgo
PrüfAlgo	----->	Spiel Logik
Maps	----->	Spiel Logik
Spiel Logik	----->	Punkte System

4 Entwicklungs- und Teststrategie

4.1 Projektplanung

Nr.	Datum	Meilenstein	Inhalt	Erfolgreich zu testen für Abschluss des Meilensteins:
1	25.05.2022	**Fertigstellung GUI & Levels (Unity)**	- GUI - [Startmenü] Levelauswahl, Optionen, Spiel fortsetzen, Level erstellen - [Ingame Menü] Statistiken ansehen, Level beenden/überspringen - [Toolbox für Editor] Wand/Block/Kreuz/Figur positionieren - Anzeige Anzahl benötigter Kisten/Kreuze - Editorobjekte (verfügbar für Spiel & Editor), Kollisionsprüfung, Design,.... - [???] Speicherfunktion (Editor, sowie Fortschritt in aktivem Level) - [Schnittstelle für andere Klassen] Form der Informationen aus Unity bzgl: - Anzahl der Kisten - Anzahl Kreuze - Anzahl Schritte - Zeitmessung	- Speicherbarkeit erstellter Level - Speicherbarkeit des Spielfortschritts -> Korrekte Bewegung der Figur im Level - Konsistenz der aus Unity ausgeleiteten Informationen
2	08.06.2022	**Fertigstellung Validitätsprüfung (Visual Studio)**	- Prüfmethoden implementieren für: - Mengenabgleich Kisten/Kreuze für Editor bzw. Lösungserkennung - Datenhaltungsklassen implementieren für: - Erfassung der Schritte / Zeit - Mengen Kisten/Kreuze - Unit-Tests	- Methode für Mengenabgleich muss für Unity nutzbare Rückmeldung "Level erfolgreich konfiguriert, bzw. Level abgeschlossen" zurückgeben -Methode für Ausgabe der Datenhaltungsklasse muss Menge an Schritten/Zeit für Unity nutzbar zurückgeben
3	20.06.2022	**Fertigstellung Spielerführung (Unity & Visual Studio)**	- Rückführung und Aufbereitung der Daten aus [2] - Rückmeldungen aus [2] für Spieler in Unity aufbereiten - Fehlermeldungen/Level beendet mit X Schritten, in X Sekunden	- Spieler bekommt im Editor & beim Spielen entsprechende Rückmeldungen zum Fortschritt bzw. zu Fehlern
4	23.06.2022	**Abgabe Einzelphase**	- Prüfung der vorgabengerechten Gestaltung des GIT-Projekts	
5	01.07.2022	**Fertigstellung Beta "Hohe Priorität"**	- Alle Komponenten zusammenführen zu lauffähigem Beta-Programm	- Programm lässt sich ohne Administratorrechte spielen - Funktionen hoher Priorität funktionieren - Das Grundspiel ist spielbar
6	13.07.2022	**Fertigstellung Beta "Mittlere Priorität"**	- Umsetzung Statistikfenster - Umsetzung Prüfung der Lösbarkeit (Editor)	- Statistiken zum Level/Allgemein können von Spieler eingesehen werden - Programm gibt Rückmeldung, ob das erstellte Level überhaupt lösbar ist
7	20.07.2022	**Beendigung Playtest**	- Dokumentation & Bewertung des Feedbacks	- Einzelfeedbacks methodisch bewertet
8	25.07.2022	**Fertigstellung Implementierung Playtest Ergebnisse**	- Umsetzung von Verbesserungsvorschlägen	- Ggf. Umsetzung hoch bewerteter Hinweise
9	27.07.2022	**Abgabe Projekt**	- Präsentation vorbereiten - GIT-Projekt "aufräumen"	

Tabelle 2: Projektplanung

4.2 Entwicklungsstrategie

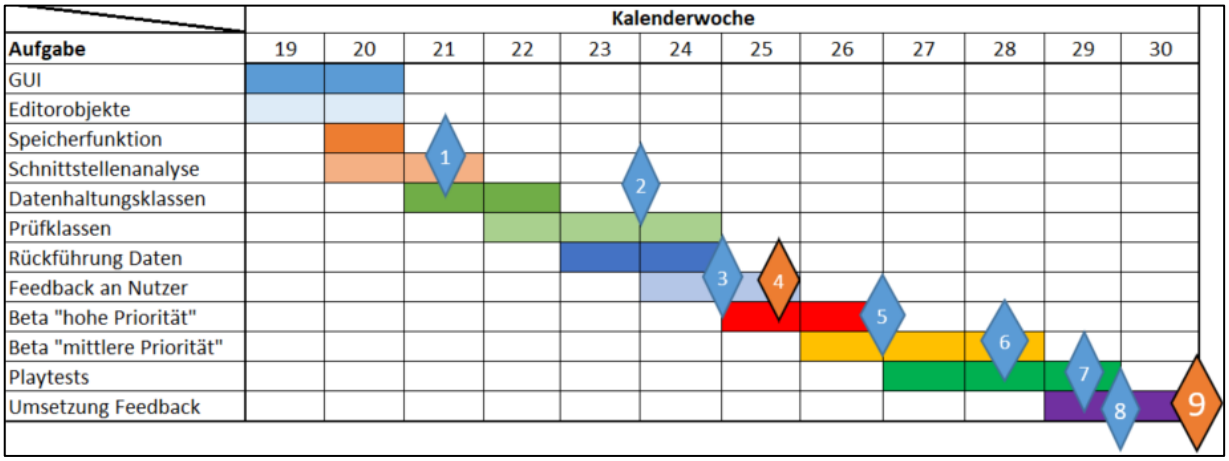


Abbildung 8: Gantt-Diagramm

Das Projekt-Gantt-Diagramm liefert zusätzlich zu der Projektplanung einen zusammenfassenden Überblick, über den Zeitverlauf der einzelnen Arbeitspakete. Er dient dem Projektteam als schneller Überblick über die Einhaltung des Zeitplans im Projektverlauf.

An jedem Meilenstein überprüfen die Projektmitarbeiter gemäß ihrer unter Kapitel 2 festgelegten Zuständigkeit, ob die Ergebnisse der Etappe die Use-Cases ihres Arbeitspakets erfüllen, oder ob in einigen Teilen nachgebessert werden muss. Hierdurch soll das Erfüllen aller Use-Cases regelmäßig geprüft werden. Dies sichert gleichzeitig das Treffen der in der Anforderungsliste dokumentierten Kundenwünsche.

Durch das Verwenden mehrerer Branches können Back-ups erstellt werden, welche es ermöglichen, problemlos Änderungen vorzunehmen, ohne ein Verschwinden des bearbeiteten Codes. Dies bietet eine gute Vorlage, um am Code zu experimentieren und möglicherweise zu optimieren. Aufgrund dessen sollen im Projekt mehrere Git Branches zur Entwicklung genutzt werden.

4.3 Teststrategie

In der Spieleentwicklung bietet Unit Testing im Vergleich zu anderen Branchen meist einen kleinen Spielraum.

Nichtsdestotrotz gibt es auch in der Spieleentwicklung bestimmte Abschnitte, die getestet werden müssen, um einen optimalen Ablauf des Endproduktes zu gewährleisten. Außerdem hilft es Probleme im Code schneller ausfindig zu machen und diese zu beheben, wodurch der Arbeitsprozess vereinfacht wird und somit effizienter gestaltet werden kann.

Basierend auf den davor geschilderten Aspekten wurde der Entschluss getroffen Unittest zu verwenden, um optimal funktionierenden Code zu schreiben. Die im Visual Studio zu implementierenden Klassen sollen so mithilfe von Dummy-Klassen, die die internen Schnittstellen simulieren, regelmäßig geprüft werden. Die Unit-Tests ermöglichen somit nach Erreichen des 2. Meilensteins eine solide Basis zur Entwicklung der restlichen Programmfunktionen und sichern auch die Funktion der bestehenden Komponenten, sollten sie im Laufe der weiteren Entwicklung noch einmal angepasst werden müssen.

Zur Prüfung des allgemeinen Spielprinzips werden nach Fertigstellung des Grundspiels, welches zunächst nur Anforderungen hoher Priorität umsetzt, Playtests durchgeführt. Durch dieses externe Feedback kann ein neuer Blick zur Bewertung der bisherigen Ergebnisse geschaffen werden, der das Finden eventueller Probleme im Spielablauf vereinfacht.

5 Lizenz

Für die Wahl der Lizenz setzt das Projektteam auf eine möglichst offene Weiterverwendung der entstehenden Software. Für die erzeugten *Projektklassen* wird hier die MIT-Lizenz favorisiert: