

# PoCs

Die PoCs dienen als Grundlage für spätere Implementationen

**Es werden PoCs benötigt zu folgenden Komponenten und ihren Sub-Elementen:**

Spieler-Character:

- Movement
- Sprite
  - Animation?
- Nähe zu Interaktionsobjekte
  - Interagieren mit Aktivitäten/Aufgaben

Raum:

- Gegenstände/Aktivitäten
- Kollision
- Sprites

Interaktionsobjekte (Aufgaben/Aktivität)

- Visuelle Effekte

Parameter:

- Manipulierbar
  - Add
  - Subtract
- Zugriff
  - Read
- Darstellung/Anzeige
  - UI
- Interaktion Untereinander
  - Multiplikator-Wert beeinflusst Basis-Wert

Zeit:

- Darstellung/Anzeige
- Verfall der Zeit
- Deadline
  - Darstellung

UI:

- Korrekte Anzeige von Parametern
- Visuelle Darstellung und Transformation von Text
- Tooltips

Visuelle Effekte

Tooltips

# Spieler-Charakter PoCs 100

## PoC 101: Player Movement

Es gilt zu beweisen, dass die X und Y Position des Spieler-Charakters mittels der Tasten WASD verändert werden kann, wobei jede der Tasten einer Erhöhung oder Verringerung der Positionsparameter hervorbringt (Bei Taste „W“ erhöht sich der Y-Positionsparameter, bei Taste „S“ verringert sich dieser).

**Exit Kriterium:** X oder Y Koordinaten der Player-Node werden bei Betätigung der WASD-Tasten verändert und der Character bewegt sich auf dem Bild.

**Fail Kriterium:** Keine Bewegung des Spieler Characters ist erkennbar und die Verwendung der WASD-Tasten erzeugt keine Veränderung der X/Y-Koordinaten.

1. Fallback: Der Spieler erhält alternative Steuerungsmöglichkeiten, wie z.B. die Pfeiltasten oder Controller.
2. Fallback: Die Interaktion kann mittels der Maus getätigt werden.

## PoC 102: Sprite Animation laden

Es gilt zu beweisen, dass für spezifische Interaktionen, wie Bewegung des Spieler-Charakters, spezifische Sprite-Animationen zuverlässig geladen werden können.

**Exit Kriterium:** Die korrekte Sprite Animation ist bei den jeweiligen Interaktionen, mit dem System, auf den Bildschirm sichtbar.

**Fail Kriterium:** Auf dem Bildschirm ist keine Animation oder eine nicht vorhergesehene Animation sichtbar.

1. Fallback: Im Ladeprozess auf korrektes Laden der Dateien warten.
2. Fallback: Es werden keine konkreten Animationen benutzt, sondern nur einfache Sprites verwendet.

## PoC 103: Interaktionsdistanz

Es gilt zu beweisen, dass Interaktionsobjekte nur innerhalb ihrer Interaktionskollisionsbox Interaktionen zulassen, wobei im Falle einer Überlappung der Interaktionskollisionsboxen nur die Interaktion ausführbar ist, welche die geringste Distanz zum Spielercharakter hat. Die Interaktionstaste könnte die „E“-Taste sein.

**Exit Kriterium:** Der Spieler kann nur innerhalb der Kollisionsbox und mit dem Interaktionsobjekt mit der geringsten Distanz interagieren.

**Fail Kriterium:** Der Spieler kann auch außerhalb der vorgegebenen Kollisionsbox mit Objekten interagieren oder mit Objekten, welche eine höhere Distanz als das nächste Interaktionsobjekt haben interagieren oder gar nicht mit dem Interaktionsobjekt interagieren.

1. Fallback: Interaktion mit dem Objekt auch mittels der linken oder rechten Maustaste zulassen, wenn der Mauszeiger sich über dem Interaktionsobjekt befindet.
2. Fallback: Interaktion nicht von Distanz, sondern von Kollision, also direkter Berührung des Spieler-Charakters mit dem Interaktionsobjekt, oder Position abhängig machen.

# Raum PoCs 200

## PoC 201: Raum Kollision

Es gilt zu beweisen, dass der Spieler Character mit der Kollisionsbox der Wand des Raumes und der in diesen enthaltenen Objekten kollidiert und diesen nicht verlassen kann.

**Exit Kriterium:** Der Spieler Character kann den Raum nicht durch das Durchgehen von Wänden verlassen.

**Fail Kriterium:** Der Spieler Character kann durch Wände gehen und es findet keine Kollision mit der Wand statt.

1. Fallback: Die X und Y Position des Character kann nur innerhalb der X und Y Werte der Größe des Raumes verwendet werden.
2. Fallback: Spieler Bewegung nur mittels der Maus erlauben und die Raumgröße auf das gesamte Fenster setzen.

# Parameter PoCs 300

## PoC 301: Parameter Manipulation

Es gilt zu beweisen, dass die Parameter Stress, Gesundheit, Produktivität und Arbeitsfortschritt bei einer Interaktion mit einem Interaktionsobjekt um den Wert X, welche dem Interaktionsobjekt zugehören, erhöht oder verringert werden können. (Bsp.: Aktivität Schlafen (Gesundheit +20, Stress -10, Produktivität +10\*Zeit, Arbeitsfortschritt + 0))

**Exit Kriterium:** Die Parameter verändern sich bei Interaktion mit einem Interaktionsobjekt und diese Veränderung ist im UI oder in der Konsole sichtbar.

**Fail Kriterium:** Die Parameter werden bei Interaktion nicht verändert, oder die Veränderung ist weder im UI noch in der Konsole erkennbar.

1. Fallback: Die Interaktion mit Objekten wird in einem Array, einer Queue oder vergleichbarem festgehalten, wobei ersichtlich ist, welche Objekte in welcher Reihenfolge verwendet wurden, was eine Berechnung der Parameter erlaubt, und somit eine Alternative zur Manipulation.

## PoC 302: Parameter Zugriff

Es gilt zu beweisen, dass die Parameter für Stress, Gesundheit, Produktivität und Arbeitsfortschritt von sich selbst und von anderen Funktionen zugreifbar und veränderbar sind.

**Exit Kriterium:** Auf die Parameter kann von innerhalb der Funktion und von anderen Funktionen zugegriffen und verändert werden.

**Fail Kriterium:** Parameter lassen sich nicht verändern und sind nicht oder nur bedingt zugreifbar.

1. Fallback: Standard-Verhalten von Objekten erzeugen.
2. Fallback: Extra-Parameter in Objekten, welche eine Berechnung des Spielerverhaltens erlauben.

## PoC 303: Konstante Addition von Parametern

Es gilt zu beweisen, dass ein veränderbare Variabel in einem regelmäßigen Intervall auf einen den Parametern addiert werden kann.

**Exit Kriterium:** Auf die Variabel kann zugegriffen und verändert werden und sie kann zuverlässig und in einem regelmäßigen abstand auf einen Parameter addiert werden.

**Fail Kriterium:** Die Variabel kann nicht, unzuverlässig oder nur in unregelmäßigen Abständen auf die Parameter addiert werden.

1. Fallback: Werte werden nur als Ganzes addiert und es gibt keine konstante Addition.

## PoC 304: Obere und untere Grenzen für Parameter

Es gilt zu beweisen, dass für die Parameter Stress, Gesundheit und Produktivität eine Ober- und Untergrenze festgelegt werden kann, welche nicht überschritten werden kann.

**Exit Kriterium:** Ober- und Untergrenzen können für die Parameter festgelegt und verändert werden und sie verhindern, dass der Parameterwert bei externen Additionen den festgelegten wert über- oder unterschreiten.

**Fail Kriterium:** Die Ober- und Untergrenzen werden dennoch überschritten oder schränken einen falschen Raum ein.

1. Fallback: Wenn der angegebene Wert überschritten wird, wird er mit diesem Wert zurück überschrieben.

## PoC 305: Produktivitätsparameter berechnen

[ToDo]

## PoC 306: Zeitparameter abrufen

Es gilt zu beweisen, dass der Parameter „Zeit“ abrufbar ist.

**Exit Kriterium:** Der Parameter „Zeit“ wird abgerufen und ist im UI oder der Konsole lesbar.

**Fail Kriterium:** Der Parameter „Zeit“ kann nicht abgerufen werden und ist wird daher nicht im UI oder der Konsole lesbar oder es ist nur ein Standardparameter lesbar.

1. Fallback: Statt Zeit zu zählen, wird jede Interaktion in einem Array oder einer Queue gespeichert, wobei jede Interaktion zugewiesen „Zeit“-Werte hat, welche die Dauer der Interaktion bestimmen.
2. Fallback: Nach einer Zeit von z.B. 600-Sekunden, endet das Spiel und der Fortschritt bis dahin wird ausgewertet.

## PoC 307: Aufzählen des Zeitparameters

Es gilt zu beweisen, dass der Zeitparameter in regelmäßigen Zeitabständen um 1 inkrementiert, und so das Voranschreiten der Zeit im Spiel simuliert.

**Exit Kriterium:** Der Parameter wird mit jedem Inkrementieren in einem regelmäßigen Zeitabstand in der Konsole ausgegeben und die Werte haben eine Differenz von 1.

**Fail Kriterium:** Die Inkrementierung, welche in der Konsole ausgegeben wird, findet nicht regelmäßig statt und die Werte haben eine Differenz größer oder kleiner als 1.

1. Fallback: Aktuelle Systemzeit in einer Variable speichern, um Zielzeit (z.B. 1000 Sekunden) addieren, und regelmäßig prüfen, ob die Systemzeit größer als die Zeit in der Variable ist.
2. Fallback: Statt Zeit zu zählen, wird jede Interaktion in einem Array oder einer Queue gespeichert, wobei jede Interaktion zugewiesen „Zeit“-Werte hat, welche die Dauer der Interaktion bestimmen.

## PoC 308: Deadline-Meilenstein

Es gilt zu beweisen, dass bei Überschreiten des Zeitwertes eines Deadline-Meilensteins (z.B. 100 Sekunden, dann 200, usw.) eine Nachricht ausgelöst wird, welche die Verbleibende Zeit anzeigt.

**Exit Kriterium:** Es wird ein Text angezeigt, welcher sagt: „Nur noch X Tage“, wobei der Wert X mit jeder Wiederholung geringer wird.

**Fail Kriterium:** Es wird kein Text angezeigt, welcher einen Hinweis auf die verbleibende Zeit gibt.

1. Fallback: Die Ausgabe wird in der Konsole getätigt, wobei die Konsole lesbar ist.

## PoC 309: Deadline

Es gilt zu beweisen, dass bei Überschreiten des Zeitwertes für die Deadline (z.B. 1000 Sekunden) das Ende des Spiels ausgelöst wird.

**Exit Kriterium:** Nach überschreiten des Zeitwertes für die Deadline wird die Endsequenz ausgelöst und der Spieler sieht diese.

**Fail Kriterium:** Nach überschreiten des Zeitwertes wird keine Endsequenz ausgelöst, oder es wird eine Endsequenz ohne Überschreitung des Zeitwertes ausgelöst.

1. Fallback: Das Ende kann nach Überschreitung des Zeitwertes manuell abgerufen werden.

## PoC 310: Korrekte Endsequenz

Es gilt zu beweisen, dass bei Überschreiten der Deadline die aufgezeigte Endsequenz anhand des Arbeitsfortschritt korrekt ausgewählt wird (Also Endsequenz 1 kommt nur wenn ein Arbeitsfortschritt von z.B. 600 erreicht wurde).

**Exit Kriterium:** Die Endsequenz wird anhand des ihr zugewiesenen Arbeitsfortschritt-Werts ausgewählt.

**Fail Kriterium:** Die Endsequenz wird nicht anhand ihres Arbeitsfortschritt-Werts ausgewählt oder es kommt zu Fehlern oder es wird keine Endsequenz gewählt.

1. Fallback: Verwendung einer Standard-Endsequenz, für andere Endsequenzen muss ein Mindestwert für den Arbeitsfortschritt-Wert überschritten werden.

# UI PoCs 400

## PoC 401: Parameter UI

Es gilt zu beweisen, dass die Darstellung der Parameter Stress, Gesundheit und Produktivität mittels des UIs die systeminternen Parameter, dargestellt durch die Konsole, abbildet und gleicht.

**Exit Kriterium:** Die vom UI dargestellten Parameter stimmen mit den in der Konsole angegebenen Werten überein.

**Fail Kriterium:** Die vom UI dargestellten Parameter stimmen nicht mit den in der Konsole angegebenen Werten überein.

1. Fallback: Die Parameter-Werte werden zusätzlich als Text dargestellt.
2. Fallback: Zusätzlich anhand von Objekten im Raum oder dem Sprite des Spieler-Charakters das UI darstellen, welche ein indirektes Lesen der Parameter erlauben.

## PoC 402: UI-Lesbarkeit (Design)

Es gilt zu beweisen, dass der Spieler (z.B. Tester oder Entwickler) das UI lesen und korrekt interpretieren kann.

**Exit Kriterium:** Der Spieler (z.B. Tester) interpretiert das UI korrekt und kann die dargestellten Werte korrekt ablesen.

**Fail Kriterium:** Der Spieler kann das UI nicht korrekt interpretieren und die Werte nicht korrekt ablesen.

1. Fallback: Alternative Darstellungsformen der Parameter anbieten, z.B. numerische Angaben zusätzlich zur Grafischen Darstellung.
2. Fallback: Abhängig von den Parametern Stress, Gesundheit oder Produktivität auch den Sprite des Spieler-Charakters verändern, um den Wert der Parameter über den Spieler Character zu vermitteln.

## PoC 403: Skalierbare Fenstergröße

Es gilt zu beweisen, dass das Fenster des Spiels auf unterschiedlichen Bildschirmgrößen korrekt skaliert und die Grafiken sich nicht verzerren oder Inhalte außerhalb der Spielwelt dargestellt werden.

**Exit Kriterium:** Das Fenster lässt sich beliebig skalieren, ohne dass sich die Grafiken verzerren, oder Inhalte außerhalb der Spielwelt sichtbar sind.

**Fail Kriterium:** Das Fenster lässt sich nicht skalieren oder die Grafiken werden nicht korrekt angezeigt oder es sind Inhalte außerhalb der Spielwelt sichtbar, welche nicht sichtbar sein sollten, z.B. das Ende einer Skybox oder eines Hintergrunds.

1. Fallback: Es wird nur eine feste Fenstergröße implementiert.
2. Fallback: Der Inhalt des Fensters behält immer dieselbe Größe unabhängig der Fenstergröße.

# Tooltips (Optional, in Aussicht)

## PoC 501: Tooltip Sichtbarkeit (Mockup noch nicht vorhanden)

Es gilt zu beweisen, dass wenn mit dem Mauszeiger über einen der im UI dargestellten Parameterwerte gehalten wird, dass ein Tooltip-UI-Element sichtbar wird, welches den Erklärungstext zum Parameter enthält.

**Exit Kriterium:** Beim Zeigen auf das UI für einen der Parameter mit dem Mauszeiger erscheint ein Tooltip, welches den Erklärungstext zum Parameter enthält.

**Fail Kriterium:** Es erscheint kein Tooltip, auch wenn der Mauszeiger über dem UI für die Parameter ist, oder es befindet sich kein Text in diesem Tooltip.

1. Fallback: Die Erklärungstexte können in einem README, welches mit dem Projekt zusammen verfügbar ist, gelesen werden.