

# Pflichtenheft

KALKULATIONSPROGRAM

FÜR

OXYGEN NOT INCLUDED

Mitwirkende: Parsa, Silvana, Vincent, Benjamin

## INHALTSVERZEICHNIS

1 Einleitung .....	x
2 Ausgangssituation und Zielsetzung .....	x
3 Dekomposition des Gesamtsystems .....	x
4 Schnittstellenübersicht .....	x
5 Lebenszyklusanalyse .....	x
6 Funktionale Anforderungen .....	x
7 Nicht-funktionale Anforderungen .....	x
8 Anforderungsverfolgung zum Lastenheft .....	x
9 Anforderungsverfolgung zu den Spezifikationen .....	x
10 Abnahmekriterien und Vorgehen zur Ausgangsprüfung .....	x
11 Lieferumfang .....	x
12 Glossar .....	x
13 Abkürzungsverzeichnis .....	x
14 Literaturverzeichnis .....	x
15 Abbildungsverzeichnis .....	x

## **1 Einleitung**

Dieses Pflichtenheft ist die Basis des Kalkulationsprogramm für das Videospiel Oxygen Not Included.

## 2 Ausgangssituation und Zielsetzung

Das Computerspiel Oxygen Not Included gehört zu den Kategorien Survival und Base-Builder. Es besitzt die Mechanik, dass Spielemente Ressourcen erstellen, umwandeln oder verbrauchen können. Diese Mechanik kann als einfache Gleichungsrechnungen (gefolgt: Rechnungen) dargestellt werden. Diese Rechnungen zu veranschaulichen und das Manipulieren zu ermöglichen, soll die Hauptaufgabe des Programms werden. Das Veranschaulichen soll als ein interaktives Wiki realisiert werden, was Teil des Programms sein soll. Drei Rechnungen folgen als Beispiel:

Unser Spielement ist der Kohlegenerator.

Die Produkte sind Strom und CO<sub>2</sub>. Der Verbrauch wird als negatives Produkt angegeben, und ist Kohle in diesem Fall.

### **Spezifischen Daten:**

Kohlegenerator: 1

Kohle: - 1kg/s

Strom: + 600W

CO<sub>2</sub>: + 0,02kg/s

Mit Hilfe des Rechner kann ich z.B.: Zeiten variieren:

### **Produkte für einen Spieltag (600s):**

Kohlegenerator: 1

Kohle: - 600kg

Strom: + 600W\*600s

CO<sub>2</sub>: + 1,2kg

### **Produkte bei 4 Kohle Generatoren:**

Kohlegenerator: 1

Kohle: - 4kg/s

Strom: + 2400W

CO<sub>2</sub>: + 0,08kg/s

### **Produkte bei gewollter Stromleistung von 1000W**

Kohlegenerator~: 1.6

Kohle~: - 1kg/s

Strom: + 1000W

CO<sub>2</sub>~: + 0,033kg/s

Das Programm soll alle Informationen die für diese Rechnungen notwendig sind, ausgeben können und für Änderungen

### **3 Dekomposition des Gesamtsystems**

Das Gesamtsystem wird in folgende Module unterteilt:

UI-/Interaktionsmodul

Eingabe von Tiles, Ressourcen, Temperaturen, Rezepten

Anzeige von Berechnungsergebnissen und Diagrammen Datenmodell

Speicherung von Spielementen, Variablen und Rezepten

Schnittstelle für Berechnungsmodul und Import/Export

Berechnungsmodul

Kernberechnungen: Produktionswerte, Energie, Ressourcenverbrauch

Szenario-Übungen – etwa mit diversen Stromquellen oder bestimmter Leistungsabgabe

Import/Export-Modul

JSON-/CSV-Export der Ergebnisse

Import von externen Daten für neue Rezepte oder Elemente

Fehler- und Validierungsmodul

Prüfung von Eingaben auf Plausibilität

Ausgabe von verständlichen Fehlermeldungen

## 4 Schnittstellenübersicht

Die Informationen aller Werte, die aus dem Spiel hervorgehen, sollen über eine Schnittstelle zwischen dem Programm und dem Quellcode des Spiels gesammelt werden.

## 5 Lebenszyklusanalyse

Das Umsetzungsmodell wird noch in der Gruppe diskutiert. Feature Driven Development soll ein Kern dieses Modells sein, um eine funktionelle Software schnell zu produzieren und sie danach mit Features zu erweitern.

Das gesamte Projekt ist über Github zugreifbar.

## 6 Funktionale Anforderungen

Gegeben sind alle Spielemente, die Variablen für ihre Produktionswerte besitzen. Alle diese Werte sollen in einer Gleichung isoliert werden können, um danach diese wie gewünscht anzupassen und alle Folgewerte auszuwerten.

## 7 Nicht-funktionale Anforderungen

### Zuverlässigkeit

Die Software sollte durch Fehleingaben nicht zum Absturz gebracht werden können.

### Effizienz

Die Berechnungen sollen nicht länger als 1 Sekunde dauern.

### Änderbarkeit

Gängige Konventionen sollen gefolgt werden.

Die Software sollte gut dokumentiert sein, sodass auch andere Teams Änderungen vornehmen können oder die Teile wiederverwenden können.

### Übertragbarkeit

Keine besonderen Vorgaben.

### Wartbarkeit

Die Wartbarkeit soll durch gut strukturierten und Kommentierten Code, sowie angemessene Dokumentation gegeben sein. Tests für die Software sollen umfangreich sein.

## **8 Anforderungsverfolgung zum Lastenheft**

## 9 Anforderungsverfolgung zu den Spezifikationen

<b>Spezifikation</b>	<b>Umsetzung im Pflichtenheft</b>
Alle Spielemente mit Variablen abbilden	Berechnungsmodul
Gleichungen isolieren und anpassen	Funktionale Anforderungen
Testbarkeit sicherstellen	Abnahmekriterien (Abschnitt 10)

## **10 ABNAHMEKRITERIEN UND VORGEHEN ZUR AUSGANGSPRÜFUNG**

- 1. Voller Zugriff auf das Github-Projekt.**
- 2. Quellcode liegt vollständig und lauffähig vor.**
- 3. Es wird mit der Gruppe ein Testkonzept entwickelt, das die Software testet.**
- 4. Die Tests laufen Fehlerfrei. Das ist automatisiert zu dokumentieren.**

## **11 Lieferumfang**

- 1. Applikation mit Code in C#**
- 2. Dokumentation über Applikation**
- 3. Testapplikation**
- 4. Dokumentation über Testapplikation**
- 5.**

## **12 Glossar (Erklärung von Fachbegriffen)**

**Tile:** Grundelement der Spielwelt, das Ressourcen enthalten kann

**Spieltag:** 600 Spielsekunden in Oxygen Not Included

**Kohlegenerator:** Spielelement zur Stromerzeugung durch Kohleverbrauch

**Wert:** Produktionsrate eines Spielobjekts pro Sekunde

## 13 Abkürzungsverzeichnis

1 **ONI**: Oxygen Not Included

2 kg: Kilogramm

3 W: Watt

4 s: Sekunde

5 JSON: JavaScript Object Notation

6 CSV: Comma-Separated Values

## **14 Literaturverzeichnis**

**ONI-Spiel, GitHub-FDD Guide, ISO Norm**

## **15 Abbildungsverzeichnis**

**1 Abbildung: Übersicht der Systemmodule**

**2 Abbildung: Datenfluss zwischen UI, Berechnungsmodul und Datenmodell**

**3 Abbildung: Beispielberechnung Kohlegenerator – Strom/CO<sub>2</sub>/Kohle**