**Requirements Analysis**

### 1. Introduction

#### 1.1. Purpose of the system

Unser System soll ein unterhaltendes 2D Videospiel aus dem Genre ACTION-ROGUELIKE, ROGUELITE werden, das in einer Welt der nordischen Mythologie spielt.

#### 1.2. Scope of the system

#### 1.3. Objectives and success criteria of the project

Unser Videospiel soll mindestens eine spielbare Welt haben, es soll mindestens einen weiblichen und einen männlichen Charakter geben, wobei einer Nahkampf und einer Fernkampf ist, mehr als 3 Gegnertypen aufweisen, mindestens 4 Bossmonster, mehr als 5 verschiedene Waffentypen.

Es soll pro Spielstunde höchstens einmal abstürzen.

#### 1.4. Definitions, acronyms, and abbreviations

#### 1.4.1 Definitions

#### 1.4.1.1 Player:

Ein Player ist die von dem User gesteuerte Entity.

#### 1.4.1.2 Monster:

Ein Monster ist eine gegnerische computergesteuerte entity, die das Ziel hat, die HP des Players auf Null zu setzen.

#### 1.4.1.3 Arena:

Die Arena ist das Feld, auf dem der Player und die Monster sich befinden und bewegen.

#### 1.4.2 Abbreviations:

AoE = Area of Effect

DoT = Damage over Time

Dmg = Damage

HP = Health Points

XP = Experience Points

#### 1.5. References

Unter <https://trello.com/b/HKFQ7We0/iap-computerspiel> sind unter dem Punkt “Design & Research" alle Referenzen zu finden.

#### 1.6. Overview

Unter dem Link <https://stormboard.com/storm/1901659/IAP> findet sich unser Storyboard, welches einen guten Überblick über das Spiel gibt.

### 2. Proposed system

#### 2.1. Overview

**SPÄTER**

#### 2.2. Functional requirements

#### 2.2.1 Arena:

Begrenztes 2D Feld auf dem sich Player und Monster bewegen und interagieren. Die Arena kann je nach Fortschritt unterschiedliche Darstellungen annehmen.

#### 2.2.2 Entity:

Eine Entity bewegt sich in der Arena und hat Stats:

#### 2.2.2.1 Entity Stats:

* MAX\_HP: Dies ist das maximale Leben einer Entity.
* HP: Die HP einer Entity ist ihr aktuelles Leben. Wenn dieser Wert auf Null gesetzt wird, stirbt die Entity. Die HP der Entity liegt zwischen der MAX\_HP und Null. Die HP wird verringert durch Damage und erhöht durch Items.
* Speed: Dieser Stat bestimmt die Geschwindigkeit, in der sich die Entities in der Arena fortbewegen können.
* Level: Das Level ist der Stat, welcher das Level festhält.
* Status: Eine Entity kann über einen bestimmten Zeitraum Statuseffekte haben. Darunter gibt es Vergiftung, Verlangsamung und Lifesteal. Ist eine Entity vergiftet, verliert die Entity kontinuierlich HP über einen bestimmten Zeitraum. Ist eine Entity verlangsamt, wird der Speed stat verringert. Hat eine Entity Lifesteal wird sie um 50% des verursachten Damage geheilt.

#### 2.2.2.2 Player:

Ein Player ist eine Entity. Ein Player ist am Leben, wenn seine HP nicht null ist. Sobald die HP eines Players endgültig auf null gesetzt wurde, ist das Spiel beendet. Ein Player kann bis zu X Items aufsammeln und diese verbessern bis maximal Stufe X anhand deines Levels.

Zusätzliche Stats:

* XP: Ein Player kann XP einsammeln um damit Level zu erhöhen. Wenn der Player zusätzlich den status XP-Multiplier hat, wird der XP-Stat um doppelt so viel erhöht.

#### 2.2.2.3 Monster:

Ein Monster ist eine Entity. Ein Monster ist ein computergesteuerter Gegner. Ein Monster kann dem Player Schaden zufügen, der anhand des Dmg Stat gemessen wird. Das Ziel eines Monsters ist es, die HP eine Spielers auf Null zu setzen. Die HP eines Monsters kann durch Player-Angriffe verringert bzw. auf Null gesetzt werden. Wenn ein Monster stirbt, droppt es XP.

Zusätzliche Stats:

* XP\_drop: Beim Sterben droppt ein Monster eine bestimmte Menge XP
* Dmg
* Attack Speed

#### 2.2.2.3.1 Boss:

Ein Boss ist ein Monster. Ein Boss ist deutlich stärker und seltener als ein gewöhnliches Monster und es droppt mehr XP. Zusätzlich kriegt man eine Belohnung in Form einer Mystic Potion und einer Trophäe.

#### 2.2.3 Damage:

Der Wert, der von den Health Points abgezogen wird, sobald die Entity von einem Angriff getroffen wird.

#### 2.2.4 Items:

Es gibt verschiedene Typen von Items (Weapon, Armor, Potion).

#### 2.2.4.1 Weapons:

Es gibt ranged, melee und Mystic Weapons. Ein Player hat n Waffenslots, kann also maximal n Waffen gleichzeitig tragen.

#### 2.2.4.1.1 Weapon Stats:

* Dmg: Jede Waffe hat ihren eigenen Damage-Wert, der einen Grundwert hat und sich außerdem dem Level des Players anpasst.
* Range: Die Reichweite des Angriffs der Waffe.
* Attack Speed: Jede Waffe hat ihren eigenen Attack Speed-Wert, der einen Grundwert hat und sich außerdem dem Level des Players anpasst.

#### 2.2.4.1.2 Ranged Weapons:

Eine Weapon wird als ranged gezählt, wenn die Range mindestens 3 beträgt. Ranged Weapons müssen vom User per Mausklick aktiviert werden.

#### 2.2.4.1.3 Melee Weapons:

Eine Weapon wird als melee gezählt, wenn die Range weniger als 3 beträgt. Melee Weapons müssen vom User per Mausklick aktiviert werden.

2.2.4.1.4 Mystic Weapons:

Mystic Weapons können beim Levelaufstieg als Waffe zur Auswahl stehen.

Mystic Weapons können beliebige Reichweite haben, sind aber generell stärkere und seltenere Items. Mystic Weapons werden periodisch aktiviert, ohne dass der User etwas tun muss. Sie können zeitgleich mit den ranged/melee Weapons genutzt werden und belegen keinen Waffenslot des Players.

#### 2.2.4.2 Armor:

Armor ist ein Item, welches bewirkt, dass der Player reduzierten Damage erhält.

#### 2.2.4.2.1 Armor Stat:

Der Armor stat legt fest, wie viel Damage reduziert wird. (Formel: …)

#### 2.2.4.3 Potions:

Potions sind eine mögliche Auswahl unter den Belohnungen für das Aufsteigen von Leveln. Diese können dem Player Statuseffekte für die Runde geben und Items verschiedene Multiplier für die Runde geben.

#### 2.2.4.4 Mystic Potions:

Mystic Potions werden von Bossen gedroppt.

* 2x XP für 60sec
* Player bekommt 50% Lifesteal 60sec
* Double Damage 60sec

#### 2.2.5 Trophäen:

Jeder Boss lässt eine spezifische Trophäe fallen. Man kann jede Trophäe aufsammeln. Diese wirken sich am Ende einer Runde auf den Highscore aus. Dazu haben sie noch einen optischen Wert und werden im Menü angezeigt.

#### 2.2.6 High Score:

Ein Score, der am Ende einer Runde basierend auf Zeit, gesammelten XP und Trophäen berechnet wird.

*% Please number all sub-items*

*% The big mistake at the beginning is that the requirements are specified far too vaguely, so the requirements should be specified as much as is possible at the current stage and they should be verifiable*

*% If a request is found to be unreachable, it will be marked as unreachable*

*% BEFORE PLANNING the TASKS in SCRUM MEETINGS, the requirement is specified in detail.*

#### 2.3. Nonfunctional requirements

* 30 fps mindestens auf einem System mit mindestens 4 GB RAM.
* Es soll pro Spielstunde höchstens einmal abstürzen.

#### 2.3.0 Coding Style:

* objektorientiert

#### 2.3.0.1 Naming rules

#### 2.3.0.1.1 Code

* Names of classes, methods, enumerations, public fields, public properties, namespaces: PascalCase.
* Names of local variables, parameters: camelCase.
* Names of private, protected, internal and protected internal fields and properties: \_camelCase.
* Names of interfaces start with I, e.g. IInterface.

#### 2.3.0.1.2 Files

* Filenames and directory names are PascalCase, e.g. MyFile.cs.
* Where possible the file name should be the same as the name of the main class in the file, e.g. MyClass.cs.
* In general, prefer one core class per file.

#### 2.3.0.2 Organization

* Class member ordering:
  + Group class members in the following order:
    - Fields and properties.
    - Constructors and finalizers.
    - Methods.

Where possible, group interface implementations together.

#### 2.3.0.3 Whitespace rules

* A maximum of one statement per line.
* A maximum of one assignment per statement.
* Indentation of 1 tab.
* A maximum of 100 characters per line.
* No line break before opening brace.
* No line break between closing brace and else.
* Braces used even when optional.
* Space after if/for/while etc., and after commas.
* No space after an opening parenthesis or before a closing parenthesis.
* One space between the operator and each operand of all other operators.

2.3.0.4 Constants

* Variables and fields that can be made const should always be made const.
* If const isn’t possible, readonly can be a suitable alternative.
* Prefer named constants to magic numbers.

### 

#### 2.3.0.5 Expression body syntax

* As with methods and other scoped blocks of code, align the closing with the first character of the line that includes the opening brace. See sample code for examples.

#### 2.3.0.6 Structs and classes

* Almost always use a class.
* Consider struct when the type can be treated like other value types - for example, if instances of the type are small and commonly short-lived or are commonly embedded in other objects. Good examples include Vector3, Quaternion and Bounds.
* Note that this guidance may vary from team to team where, for example, performance issues might force the use of structs.

#### 2.3.0.7 Array vs List

* In general, prefer List<> over arrays for public variables, properties, and return types (keeping in mind the guidance on IList / IEnumerable / IReadOnlyList above).
* Prefer List<> when the size of the container can change.
* Prefer arrays when the size of the container is fixed and known at construction time.
* Prefer array for multidimensional arrays.

#### 2.3.0.8 Folders and file locations

* Be consistent with the project.
* Prefer an organized structure where possible. e.g. /MyProject/Assets/Player

#### 2.3.0.9 Use of tuple as a return type

* In general, prefer a named class type over Tuple<>, particularly when returning complex types.

#### 2.3.0.10 Object Initializer syntax

For example:

var x = new SomeClass {

Property1 = value1,

Property2 = value2,

};

* Object Initializer Syntax is fine for ‘plain old data’ types.
* Avoid using this syntax for classes or structs with constructors.
* If splitting across multiple lines, indent one block level.

#### 2.3.0.11 The var keyword

* Use of var is encouraged if it aids readability by avoiding type names that are noisy, obvious, or unimportant.
* Encouraged:  
  + When the type is obvious - e.g. var apple = new Apple();, or var request = Factory.Create<HttpRequest>();
  + For transient variables that are only passed directly to other methods - e.g. var item = GetItem(); ProcessItem(item);
* Discouraged:  
  + When working with basic types - e.g. var success = true;
  + When working with compiler-resolved built-in numeric types - e.g. var number = 12 \* ReturnsFloat();
  + When users would clearly benefit from knowing the type - e.g. var listOfItems = GetList();

#### 2.3.0.12 Attributes

* Attributes should appear on the line above the field, property, or method they are associated with, separated from the member by a newline.
* Multiple attributes should be separated by newlines. This allows for easier adding and removing of attributes, and ensures each attribute is easy to search for.

% Please include the coding style here, as well as the performance as the frame rate that is to be achieved

% the requirements are written in such a way that they can also be checked - later this should also be checked explicitly in the results

#### 2.3.1. User interface and human factors

Der User muss eine Tastatur und Maus benutzen, um mit dem Spiel zu interagieren. Dabei werden standardmäßig die linke und rechte Maustaste benutzt, so wie WASD oder die Pfeiltasten.

Vor dem Spiel kann der Spieler durch ein Startmenü navigieren, in dem er/sie das Spiel starten kann.

Während des Spiels sollen mindestens die aktiven Items und der Score angezeigt werden.

#### 2.3.2. Documentation

Der Entwicklungsprozess wird auf [Trello](https://trello.com/b/HKFQ7We0/iap-computerspiel) und Git dokumentiert.

#### 2.3.3. Error handling and extreme conditions

Wenn das Spiel abstürzt, wird ein error log erstellt, der die zuletzt ausgeführten Funktionen angibt, um das Problem zu erkennen.

#### 2.3.4. Physical environment

Das Spiel muss auf einem Computer installiert werden, welcher als Betriebssystem Windows 10 oder neuer bzw. MacOS 13.1 oder neuer hat. Dieser muss Zugriff auf einen Bildschirm haben.

#### 2.3.5. Resource issues

In den Optionen gibt es die Möglichkeit, die maximale Anzahl an FPS einzustellen.

#### 

#### 2.4. Pseudo requirements (???)

#### 2.5. System models

#### 2.5.1. Scenarios (???)

#### 

#### 2.5.2. Use case model

Zugriff über [Trello](https://trello.com/b/HKFQ7We0/iap-computerspiel).( Unter Design)

#### 2.5.3. Object model

#### 

#### 2.5.3.1. Data dictionary (???)

#### 

#### 2.5.3.2. Class diagrams

Klassen Diagram ist in [Trello](https://trello.com/b/HKFQ7We0/iap-computerspiel) unter Design und Research aufzufinden.(Klassen-Diagramm)

#### 

#### 2.5.4. Dynamic models(???)

#### 

#### 2.5.5. User-interface -- navigational paths and screen mock-ups

Mockups sind in [Trello](https://trello.com/b/HKFQ7We0/iap-computerspiel) unter Design und Research aufzufinden.

#### 

### 3. Glossary