# Teilnehmer/innen des Teams:

|  |  |
| --- | --- |
| Klasse:  BI16c | Team:  Lukas Hunziker, Kavinth Murugathas |

# Anforderungsdefinition (Meilenstein A)

|  |  |
| --- | --- |
| „Dungeon’Out“ | |
| **Auftrag:**  (Allgemeine Beschreibung) | Kundenauftrag: Der Kunde möchte ein neues Dungeon Game. Der Spieler muss sich durch verschiedene Gegner durchsetzen, um auf ein neues Level zu steigen. Der Spieler hat die Fähigkeit Projektile zu verschiessen und hat eine Lebensanzeige. Der Spieler verliert Leben, wenn er in Kontakt kommt mit dem Gegner. Die Bewegungen der Gegner sind zufällig. Unterschiedliche Gegner haben unterschiedliche Fähigkeiten, es gibt Nahkampf- und Fernkampftypen. Falls der Spieler alle Feinde eliminiert hat muss er das Tor am oberen Rand passieren um auf ein neues Level aufzusteigen. Nach einer gewissen Anzahl an Levels kommt ein Bossfight.  Ausserdem Durch das Eliminieren von Feinden verdient der Gamer XP. Mit der XP kann man beim Aufsteigen auf ein neues Level zwischen drei Fähigkeiten auswählen und kaufen.  **Userinput:**   * User kann Projektile abschiessen * Tor passieren, um Level aufzusteigen   Ein Bild, das Shoji enthält.  Automatisch generierte Beschreibung  **Skizze 🡪**   * Spieler und Projektil * Tor zur neuen Welt * Verschiedene Gegner |
| **MUSS**  **Kriterien:**  (Konkrete Features, die umzusetzen sind) | **Ablauf des Scripts:**   * Spieler und Gegner spawnen * Spieler schiesst Projektile * Gegner abschiessen und verschwinden lassen * Spieler und Projektil können nicht durch Wände gehen * Spieler verliert Leben, falls er in Berührung mit dem Feind kommt * Aufsteigen auf ein neues Level erst wenn alle Gegner eliminiert worden sind |
| **KANN**  **Kriterien:**  (Konkrete Features, die optional sind) | **Zusatz:**   * Nach der Eliminierung eines Gegners verdient man XP * womit man beim Aufsteigen auf ein neues Level Items kaufen kann * Verschiedene Gegner-Objekte |

## Planung LB2

|  |  |  |  |
| --- | --- | --- | --- |
| *MS* | *Tätigkeit / Abgabe* | *Soll-Datum* | *Ist-Datum* |
| A | ProjektstartTeam BildungWahl / Ausarbeitung der Anforderungsdefinition Abnahme Anforderungsdefinition durch Lehrperson |  |  |
| B | Teamaufgabe 1:Abgabe: Lösungsdesign  (Analyse, Design: Funktionsmodell, UseCase, GUI, Storyboard) |  |  |
|  |  |  |  |
| B2 | Teamaufgabe 2:Abgabe: Testvorschrift und Testfälle |  |  |
|  |  |  |  |
| C | Einzelaufgabe 3:Abgabe Szenario (.zip) mit Inline-Dokumentation, Systemdokumentation (UML Klassen-, Sequenzdiagramm)Fachgespräch Projektabnahme |  |  |
| C2 | Einzelaufgabe 4:Abgabe: Ausgefüllter Systemtest |  |  |

# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

## Funktionsmodell

Im Folgenden sind die erwarteten Eingaben und Ausgaben beschrieben / dargestellt:

...

(Funktionsmodell)

Legende:

* ....

## Anwendungsfälle (UseCases)

Folgende Anwendungsfälle sind hier detailliert dokumentiert:

...

(Detaillierte Beschreibung der UseCases)

Legende:

...

## Ablauf

Aus Benutzersicht ist folgender Ablauf des Programms zu erwarten:

...

(Storyboard)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Testvorschrift (LB2 Meilenstein B2: Teamaufgabe 2)

Testbeschrieb und vorbereitetes Testprotokoll siehe Dokument   
***M226B\_LB2\_Testvorschrift\_MS-B2.docx***

# Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

Das erstellte Java-Projekt (Greenfoor-Szenario) ist hier detailliert abgelegt:

***M226B\_ Aufgabe\_3\_Szenario\_IhrName.zip***

## Statisches Design: Klassendiagramm

Folgend die statische Struktur des Szenarios

...

(UML Klassendiagramm mit Assoziationen und Kardinalitäten)

## Umfang / Abgrenzung / Änderungen gegenüber Design

Aufgrund unten beschriebener Umstände sind Anpassungen des ursprünglichen Lösungsdesigns gemacht worden:

...

(Umstände / Anpassungen / Veränderungen)

## Funktionalität der Implementation.

Zusätzlich zu der Inline-Dokumentation sind hier folgende Funktionen detailliert beschrieben:

...

(Ausführliche Beschreibung der internen Funktionen   
oder Verweis zum Inline-Kommentar mit JavaDoc! (/\*\* @param @return \*\*/)

## Dynamische Struktur: Sequenzdiagramm

Ein zentraler Ablauf eines UseCases ist im Folgenden dargestellt:

...

(Darstellung eines zentralen Ablaufs mittels Sequenzdiagramm)

**Trace: ...**

*...*

# Bedienungsanleitung (Meilenstein C: individuelle Aufgabe 3)

...

# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

Ausgefülltes Testprotokoll siehe Dokument   
***M226B\_LB2\_Testvorschrift\_MS-C2\_Name.docx***