

|  |  |
| --- | --- |
| Status | Zur Verifizierung freigegeben |
| Name des Projektes | Rocket |
| Projektleitung | J. Eckerle |
| Projektauftraggeber | J. Eckerle |
| Autoren (alphabetisch) | M. Käser, F. Schwab, M. Tschanz |
| Initialen | kasem5, schwf5, tschm23 |
| Genehmigung durch | R. Gasenzer |

Versionen

| Version | Datum | Beschrieb | Autoren |
| --- | --- | --- | --- |
| 1.0 | 13.10.2014 | Initialdokument fertig gestellt | kasem5, schwf5, tschm23 |
| 1.1 | 16.10.2014 | Änderungen im Bereich der Anforderungen | kasem5, schwf5, tschm23 |
| 1.2 | 20.10.2014 | Erweiterung durch Aufgabe 2 | kasem5, schwf5, tschm23 |
| 1.21 | 29.10.2014 | Überarbeitung, div. Ergänzungen | kasem5, schwf5, tschm23 |
| 1.3 | 05.11.2014 | Erw. von: Haupt-/Nebenzielen, Usecases | kasem5, schwf5, tschm23 |
| 1.4 | 12.11.2014 | Überarbeitung gemäss Meeting mit R. Gasenzer | kasem5, schwf5, tschm23 |
| 1.4.1 | 20.11.2014 | Überarbeitung der Anforderungen | kasem5, schwf5, tschm23 |
| 1.4.2 | 23.11.2014 | Überarbeitung der Anforderungen | kasem5, schwf5, tschm23 |
| 1.4.3 | 24.11.2014 | Überarbeitung der Anforderungen | kasem5, schwf5, tschm23 |
| 1.4.4 | 25.11.2014 | Überarbeitung der Anforderungen | kasem5, schwf5, tschm23 |

Table of Contents

[1 Einleitung 4](#_Toc404704041)

[1.1 Ziel des Dokuments 4](#_Toc404704042)

[1.2 Leserkreis des Dokuments 4](#_Toc404704043)

[2 Projektbeschrieb 4](#_Toc404704044)

[2.1 Stakeholderliste 5](#_Toc404704045)

[2.2 Nutzer- und Zielgruppen 5](#_Toc404704046)

[2.3 Projektmethode 5](#_Toc404704047)

[2.4 Technische Ressourcen 5](#_Toc404704048)

[2.5 Dokumente 5](#_Toc404704049)

[3 Projektziele 6](#_Toc404704050)

[3.1 Name des Projekts und Hauptziel (HZ1) 6](#_Toc404704051)

[3.2 Teilziele (TZ1-8) 7](#_Toc404704052)

[4 Scoping 8](#_Toc404704053)

[4.1 Rahmenbedingungen (RB) 8](#_Toc404704054)

[4.1.1 Technische Rahmenbedingungen RB1-6 8](#_Toc404704055)

[4.2 Systemkontext und Systemgrenzen 8](#_Toc404704056)

[4.3 Out of scope 9](#_Toc404704057)

[5 Anforderungen 10](#_Toc404704058)

[5.1 Quellen und Herkunft 10](#_Toc404704059)

[5.2 Anforderungsliste 10](#_Toc404704060)

[5.3 Funktionale Anforderungen 11](#_Toc404704061)

[5.3.1 Detailbeschreibung der funktionalen Anforderungen 12](#_Toc404704062)

[5.4 Qualitätsanforderungen (Nichtfunktionale Anforderungen) 15](#_Toc404704063)

[5.4.1 Detailbeschreibung der nicht-funktionalen Anforderungen 15](#_Toc404704064)

[6 Chancen und Risiken des Projekts 16](#_Toc404704065)

[7 Rollenkonzept 17](#_Toc404704066)

[8 Glossar 18](#_Toc404704067)

[8.1 Erklärungen und Übersetzungen 18](#_Toc404704068)

[8.2 GUI 19](#_Toc404704069)

[8.3 Synonyme 19](#_Toc404704070)

[9 Referenzen 19](#_Toc404704071)

**Überarbeiten der bestehenden Inhalte (nebst Schablone):**

* Anforderungen beschreiben Funktionen mit denen die Ziele erreicht werden können.   
  WER stellt WARUM die Anforderung? Beim WER die Rolle erwähnen und nicht die Person
* Mit muss, soll, kann/darf schon die Wichtigkeit andeuten.
* Anforderungen messbar formulieren, wenn nötig schreiben wie sie das Gewünschte erreichen können.
* Ergänzen zu den FR/NFR können technische Requirements aufgenommen werden
* Nominalisierung OK
* Substantive ohne Bezugsindex – wurden sie erklärt, genau definiert?
* Universalquantoren NOK - Mengen und Häufigkeiten quantifizieren OK
* Unvollständig spezifizierte Bedingungen (if-then nicht ausgearbeitet, nicht auf  
  die Alternativen und anderen Möglichkeiten eingegangen)
* Unvollständig spezifizierte Prozesswörter – wenn wir „übertragen“ verwenden,  
  müssen wir spezifizieren „was“, „wohin“ und „wie“ etwas übertragen wird +
* Funktionale Req.: Wie wird die Aktivität durchgeführt? Durch:  
   Usersteuerung -> Das System muss die Möglichkeit bieten <wem?> ...dies und das zu machen

System(autonom) -> Das System sollte/muss/wird: ...  
Fremde Systeme -> Das System wird fähig sein / wird durch...

# Einleitung

Das zu realisierende Spiel mit dem Namen „Rocket“ und die in diesem Zusammenhang entstehenden Artefakte müssen den Anforderungen des Projekts gerecht werden.

Die Mitarbeiter am Projekt kommunizieren deshalb fortgehend mit dem Projektverantwortlichen, um die notwendigen Informationen und Anforderungen elaborieren zu können. Auf der Gegenseite sind die Mitarbeiter verpflichtet, die Vorgaben originalgetreu umzusetzen.

## Ziel des Dokuments

Die Ziele und Anforderungen an das Projekt, an die einzusetzenden Technologie und die Hilfsmittel müssen vorgängig dokumentiert werden. Im Verlaufe des Projektes wird man so erkennen können, ob man auf dem richtigen Weg ist und kann abschliessend auch eine sinnvolle Evaluierung durchführen.

## Leserkreis des Dokuments

Das vorliegende Dokument steht der Öffentlichkeit zur freien Einsicht zur Verfügung. Primär richtet es sich an die Dozierenden der Module BTI7082q und BTI7301p, im Rahmen des fünften Semesters der q-Klasse an der BFH in Bern.

# Projektbeschrieb

**Zielsetzung** Ein 3D-Adventure-Spiel mit der Unity-Game-Engine entwickeln

**Technologien**  Unity Game Engine, Blender, Java-Script / UnityScript, Automaten

**Zeitressourcen** 24 Stunden geführter Unterricht, 216 Stunden Selbststudium

Entwicklung einer 3D-Adventure-Game Basis (Minimum an Levels & Funktionen) mit Hilfe der Unity3D Engine. Der Spielzustand und der Zustand der Agenten soll durch Automaten modelliert werden. Das Verhalten der Agenten resp. deren Intelligenz wird über die JavaScript nahe Sprache UnityScript realisiert. Die Agenten passen sich der jeweiligen Spielsituation an. Ein Aufbau von Wissen im Bereich der Automatentheorie ist dafür Voraussetzung.

Für die Spielidee und das Konzept wird dem Team freie Hand gelassen. Rücksprache mit dem Projektleitung bzw. mit dem Projektauftraggeber garantiert, dass die Spielidee und das Konzept den Erwartungen und Vorstellung gerecht werden.

Im Moment (Stand KW: 45) sind noch einige Zielvorstellungen und Anforderungen in Abklärung und müssen noch genauer mit dem Stakeholder geklärt werden.

## Stakeholderliste

* J. Eckerle als leitender Dozent und somit Projektauftraggeber
* Mitarbeitende am Projekt Rocket, namentlich:   
  Martin Käser  
  Fabian Schwab  
  Marcel Tschanz

## Nutzer- und Zielgruppen

* Testpersonen
* Spieler von Computerspielen und Interessierte an unserem Projekt als operative Anwender

## Projektmethode

* Das Projekt wird anlehnend an SCRUM agil entwickelt.

## Technische Ressourcen

* Unity Editor (Entwicklungsumgebung)
* UnityScript, Monodevelop
* Automatentheorie („Artificial Intelligence for Games“)
* Blender (Gestaltung 3D Modelle)
* GitHub (FileShare und Sicherheitskonzept für Projekt)
* Laptops der Projektmitglieder (Testen und Anwendung des Adventures)

## Dokumente

* Projektdokumentation
* Anforderungsdokument
* Projektzeitplan
* Back- / Sprintlog mit den untereinander abhängigen Tasks
* Diary (Kurzbeschrieb der Arbeitsaufwände)
* Arbeitsjournal (Ausführlichere Informationen zu den Tasks)
* Code Repository

# Projektziele

## Name des Projekts und Hauptziel (HZ1)

**Projekt: „ROCKET“**

HZ1: Eine spielbare *Alpha-Version*\* eines Adventuregames, dessen *Basislevel*\*und die implementierte, künstliche Intelligenz der Gegner durch selbstgeschriebene Spielklassen offen für Erweiterungen in Komplexität und Umfang bleiben. Eine Kurzgeschichte begleitet den Spieler bei seinen Handlungen und sorgt für zusätzliche Unterhaltung.

\*Glossareintrag

## Teilziele (TZ1-8)

SMART formulieren – Punkte abdecken, WAS das System leisten soll, aber nicht wie:

|  |  |
| --- | --- |
| **Teilziel-code** | Release: Alpha-Version |
| **TZ1** | Die Klassenhierarchie ermöglicht den einfachen Ausbau des Basislevels durch zusätzliche Spielkomponenten wie Räume und Gegenstände und kann durch einen Aussenstehenden mit Erfahrung im Umgang mit Unity ohne zusätzliche Einführung implementiert werden. |
|  |  |
| **TZ2** | Das Spiel enthält mindestens einen nicht spielbaren Gegner, der auf den Ausgang des Spiels Einfluss nimmt, indem er den Spieler aufsucht und am Verlassen des Levels hindert. |
|  |  |
| **TZ3** | Die Zahlenkombinationen der Rätsel innerhalb des Spiels und das Verhalten der Gegner ändern bei jedem Level-Neustart. So wird das Basislevel auch für Spieler interessant bleiben, die das Level bereits einmal absolviert haben. |
|  |  |
| **TZ4** | Durch vom Spieler ausgeführte Aktionen in einzelnen Räumen des Levels werden andere Bereiche freigeschaltet, so dass eine Abhängigkeit zwischen den Einrichtungen besteht. Um den Spielerfolg zu garantieren ist der Besuch jeder dieser Räume notwendig. |
|  |  |
| **TZ5** | Die nicht sichtbare, aber menschliche Spielfigur bewegt sich aufrecht gehend durch das Level. Das Sichtfeld entspricht dem Sichtwinkel einer aufrecht gehenden Person. Und als Spieler besitzt man die Möglichkeit, seine Figur in zwei verschiedenen Geschwindigkeiten fortzubewegen. |
|  |  |
| **TZ6** | Dank Lade-und Speichermöglichkeiten kann ein Spiel zu verschiedenen Zeitpunkten unterbrochen und wieder aufgenommen werden. Das ermöglicht auch eine komplexe Gestaltung des Levels, weil so nicht auf die dem Spieler verfügbare Zeit Rücksicht genommen werden muss. |
|  |  |
| **TZ7** | Der Spieler hat die Möglichkeit ausgezeichnete Gegenstände, die in seiner Reichweite liegen an sich zu nehmen und in seinem Inventar abzulegen. Das Spielinventar kann jederzeit durchsucht werden. Ein bestimmter Gegenstand ist von zentraler Bedeutung und wird für das Verlassen des Levels benötigt. |
|  |  |
| **TZ8** | Die Integrierung von Waffen für Spieler und Gegner ist möglich, ohne an der Klassenhierarchie Änderungen vornehmen zu müssen. |
|  |  |
| **TZ9** | Die Spielstatistik (Dauer und erreichte Punkte) wird nach dem Spielabschluss an eine zentrale Stelle gesendet. Es wird eine Gesamtstatistik verfügbar sein, damit der Spieler sich mit anderen Spielern indirekt messen und seine erreichte Punktzahl mit fremden Werten vergleichen kann. |

# Scoping

## Rahmenbedingungen (RB)

### Technische Rahmenbedingungen RB1-6

|  |  |
| --- | --- |
| **RB1** | Als Programmiersprache ist die objektorientierte Sprache C# zu verwenden |
| **RB2** | Zielplattformen sind Windows (ab Windows 7) und MacOS (ab OSX) |
| **RB3** | Als Laufzeit-und Entwicklungsumgebung ist das Produkt „Unity 3D“ zu verwenden |
| **RB4** | Als Entwicklungsumgebung ist der Editor „Mono-Develop“ zu verwenden |
| **RB5** | Zustände der Agenten müssen mit Zustandsmaschinen (hierarchisch oder einfach endlich) modelliert werden und müssen sich gegenseitig beeinflussen |
| **RB6** | Die Spielkomponenten kommunizieren untereinander über ein Messaging System, wie es für Spiele üblich ist und in *Buckland (Referenz, ID 2)* ab S. 69 beschrieben wird. |

## Systemkontext und Systemgrenzen

Grobe Architektur des 3-D-Adventures Rocket



## Out of scope

Der Fokus liegt nicht auf der Entwicklung eigener Grafiken, 3D-Modellen oder Audiodateien. Das Spiel soll ausserdem keine Rennsimulation werden. Auch ein Shooterspiel ist nicht Ziel der Entwicklung. Das Spiel enthält kein Mehrspielermodus und kann nicht über das Netzwerk oder über das Internet gespielt werden.

Die Skripte zu den Grundfunktionalitäten was Bewegungen der Spielfigur betreffen werden nicht direkt verändert. An der Konfiguration des Windows- oder Mac-Clients, auf dem das Spiel gespielt wird werden keine Änderungen vorgenommen.

# Anforderungen

## Quellen und Herkunft

JE: Projektauftraggeber Jürgen Eckerle

Team: Projektmitarbeiter M. Käser, F. Schwab, M. Tschanz im Plenum

Wo das Team als Quelle angegeben wird, ist dies als Artefakt der Gruppenarbeit zu interpretieren und wurde jeweils von allen Mitgliedern und dem Projektauftraggeber validiert.

## Anforderungslisten

### Legende und ergänzende Hinweise

**Nr.**

Die Kennung der Anforderung, die im restlichen Dokument als Referenzcode dient.

**Kurzbezeichnung**

Ein grober Beschrieb der Anforderung. Wird in der dazugehörigen Detailbeschreibung jeweils genauer erläutert.

**Status**

In welchem Zustand sich die Anforderung befindet.   
Mögliche Zustände sind: offen, geplant, in Arbeit, erledigt

**P - Priorität** Bewertung: 1(low), 2(medium), 3(high)

**V= Variabilität** Bewertung: 1(low), 2(medium), 3(high)

**K= Komplexität** Bewertung: 1(low), 2(medium), 3(high)

**R= Risiko** Bewertung : <8(low), 8-10(medium), >10(high)

Das aus {P;V;K} berechnete Risiko (Zahlenwert), ergibt sich durch Addition der gewichteten Eigenschaften Priorität (P), Variabilität (V) und Komplexität (K).

Zur Gewichtung der Eigenschaften hinsichtlich der Risikoabschätzung:

Priorität \*2, Komplexität \*2, Variabilität \*1

Priorität und Komplexität werden also doppelt gewichtet.

**Quelle**

Aus welcher Quelle (Punkt 5.1) die Anforderung entstammt.

**Datum**

Wann dieses Requirement aufgenommen wurde.

**Kursive Formatierung**

Fachausdrücke, die im Glossar näher ausgeführt wurden, sind kursiv formatiert.

## Funktionale Anforderungen

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nr.** | **Kurzbezeichnung** | **Status** | **P** | **V** | **K** | **R** | **Quelle** | **Datum** |
| FR1 | Allgemeines |  | | | | | | |
| FR1.1 | Startsequenz bei Spielstart | geplant | 3 | 1 | 1 | med | Team | 01.10.2014 |
| FR1.2 | Briefing des Spielers | geplant | 3 | 1 | 1 | med | Team | 01.10.2014 |
| FR2 | Level-Design | Bezug zu Teilziel TZ4 | | | | | | |
| FR2.1 | Ein spielbares, durch Wände abgegrenztes Level mit fixem Start und fixem End-Punkt | geplant | 3 | 3 | 2 | high | Team | 24.09.2014 |
| FR3 | GUI / Menus | Bezug zu Teilzielen TZ6, TZ9 | | | | | | |
| FR3.1 | Spielsteuerung (Tastenbelegung) an eigene Bedürfnisse anpassen können | geplant | 2 | 2 | 1 | med | Team | 01.10.2014 |
| FR3.2 | Speichern der Spielstände | erledigt | 1 | 2 | 2 | med | Team | 24.10.2014 |
| FR3.3 | Laden der Spielstände | erledigt | 1 | 2 | 2 | med | Team | 24.10.2014 |
| FR3.4 | Endsequenz mit anschliessender Statistik | geplant | 2 | 3 | 2 | high | Team | 08.10.2014 |
| FR4 | Spielfigur | Bezug zu Teilziel TZ5 | | | | | | |
| FR4.1 | Erweiterter Sichtradius durch Spezialkamera:  „Um-die-Ecke-Sicht“ | geplant | 3 | 2 | 2 | high | Team | 10.10.2014 |
| FR4.2 | Aufnahme von Gegenständen in das Inventar | geplant | 3 | 1 | 2 | high | Team | 10.10.2014 |
| FR5 | Gegner (Roboter) | Bezug zu Teilziel TZ2 | | | | | | |
| FR5.1 | Der Robotergegner patrouilliert durch das Level | erledigt | 3 | 2 | 3 | high | Team | 03.10.2014 |
| FR.5.2 | Der Robotergegner löst einen Alarm aus, sobald er den Spieler entdeckt | geplant | 1 | 3 | 3 | high | Team | 03.10.2014 |

### 

### Detailbeschreibung der funktionalen Anforderungen

Kursiv formatierte Wörter oder Wortfolgen sind im Glossar näher erklärt.

|  |  |
| --- | --- |
|  | **Detaillierte Beschreibung der funktionalen Anforderungen** |
| FR1 | Allgemeines |
| FR1.1 | **Startsequenz:** Der Aufruf der game.exe startet das Spiel „Rocket“, und das System initiert Startsequenz, welche dem Benutzer am Bildschirm präsentiert wird. Diese Startsequenz blendet während drei (3) Sekunden nebst dem Spielelogo auch den Titel des Spiels auf dem Bildschirm ein. Auf den Ablauf der Sequenz kann der Spieler keinen Einfluss nehmen. Nach Ablauf der drei (3) Sekunden wechselt das System ins Hauptmenu des Spiels wo dem Spieler die Optionen „New Game“,Load Game“ und „Control“ und „Quit“ als Auswahlmöglichkeiten zur Verfügung. |
| FR1.2 | **Briefing des Spielers:** Hat der Spieler die Option „New Game“ aus dem Hauptmenu ausgewählt, stellt das System einen Text auf dem Bildschirm dar, welcher den Spieler über das zu erreichende Ziel des *Basislevels* informiert (=Briefing). Dieses Text-Briefing darf in seiner Länge eine Bildschirmseite nicht überschreiten.  Der Spieler sollte das Briefing sofort verlassen können, um anschliessend ins Level einzusteigen. Dies für den Fall, dass der Spieler bereits mit dem Spiel vertraut ist. |
| FR2 | Level-Design |
| FR2.1 | **Spielbares, durch Wände abgegrenztes Level:** Die spielbare Basisversion des Spiels umfasst ein einzelnes Level. Dieses durch Wände begrenzte Level erstreckt sich über eine einzige Ebene. Das Level muss dem Spieler die Möglichkeit bieten, in mindestens drei verschiedene Räume mit seiner Spielfigur einzutreten. Dabei sollten sich die Räume gegenseitig im Verhalten beeinflussen. Vom Stakeholder J. Eckerle ist bezüglich der Räume eine solche minimale Komplexität gefordert. Beispiel: Schalter 1 aus Raum A öffnet Türe zu Raum B.  Der Spieler beginnt das Level bei jedem Neustart des Spiels vom selben Startpunkt aus. Auch der Ort, an dem die Spielfigur das Level verlässt soll einzigartig sein. |
| FR3 | GUI / Menus |
| FR3.1 | **Spielsteuerung manuell zu konfigurieren:** Im *Hauptmenu des Spiels* soll der Spieler die Möglichkeit haben, die Tastenbelegung für die Spielsteuerung seinen Wünschen anzupassen (Option: “Control“). Er kann bei Bedarf jeder Bewegungsmöglichkeit der Spielfigur eine beliebige Taste zuweisen. Eine Taste kann dabei aber nur eine Funktion erfüllen und wird nur einmal zu belegen sein. Das System weist den Spieler auf mehrfach eingesetzte Tasten und nicht zugeordnete Aktionen hin.  Innerhalb des Spiels sind die Einstellungen zur Spielsteuerung nicht mehr zu ändern. Als Eingabemedium ist nur die Tastatur vorgesehen. |
| FR3.2 | **Speichern der Spielstände:** Über das *Pausenmenu* des Spiels muss der Spieler die Möglichkeit besitzen, das aktuelle Spiel zu speichern. Zur Speicherung stellt das Spiel 10 Speichereinheiten zur Verfügung. Dies bedeutet, dass pro Spieler höchstens 10 verschiedene Spielstände vorhanden sein können. Sind diese 10 Speicherplätze belegt, muss ein bestehender Eintrag überschrieben werden. Ein einzelner Speichervorgang soll nicht mehr als zwei Sekunden in Anspruch nehmen. |
| FR3.3 | **Laden der Spielstände:** Um gespeicherte Spielstände zu laden muss dem Spieler im Hauptmenu und dem Pausenmenu die Möglichkeit zum Laden eines gespeicherten Spielstandes zur Verfügung stehen. Wird eine Auswahl über „Load Game“ vom Spieler getroffen lädt das System den gespeicherten Zustand.  3.31) Aus dem Hauptmenu wird der Spielstand umgehend geladen und die Spielfigur befindet sich an der Position zum Zeitpunkt des Speichervorganges.  3.32)  Aus dem Pausenmenu wird das System den Spieler fragen, ob er das aktuelle Spiel speichern will, bevor ein Spielstand geladen wird. Bei Bestätigung wird vor dem Ladevorgang nun ein neuer Eintrag in die Liste der gespeicherten Spielstände gemacht. Wird die Frage verneint, lädt das verhält sich das System analog 3.31 |
| FR3.4 | **Endsequenz und Statistik:** Erreicht der Spieler mit seiner Figur das Ende des Spiels unterbricht der Spielfluss und es wird nicht weiter auf Eingaben des Spielers reagiert. Eine Nachricht informiert den Spieler darüber, dass er das Level erfolgreich abgeschlossen hat. Auf dem Bildschirm erscheint im direkten Anschluss eine kurze Film- oder Bildsequenz, die ihr Ende in einer tabellarisch dargestellten Statistik zur erreichten Punktzahl des Durchgangs findet. Sobald die Statistik eingeblendet wird, muss der Spieler die Möglichkeit haben auszuwählen, ob der das Level neu starten oder das Spiel beenden möchte. Für beide Optionen wird ihm am Bildschirm eine entsprechende Option zur Verfügung gestellt. Wählt er keine der Optionen, bleibt der Statistikbildschirm eingeblendet. |
| FR4 | Spielfigur |
| FR4.1 | **Erweiterter Sichtradius „Sicht um die Ecke“:** Der Spieler muss die Möglichkeit haben, via Tasteneingabe mit seiner Spielfigur einen Blick „um-die-Ecke“ zu werfen. Diese Funktion dient dazu, Gegner auszumachen und/oder sich einen sicheren Überblick über den weiteren Verlauf der Spielewelt zu verschaffen. Diese Steuerungsfunktion bewirkt im eigentlichen Sinne ein Verschieben des Sichtfeldes der Spielfigur für 2-3 Sekunden nach links oder nach rechts (abhängig von der gedrückten Taste). In der Standardkonfiguration tätigt die Spielfigur den Blick nach rechts über die Taste R und den Blick nach links über die Taste Q.  Wird also eine mit dieser Sicht-Funktion belegten Taste gedrückt, verschiebt sich das Sichtfeld automatisch um eine noch zu definierende Anzahl Einheiten nach links rsp. rechts und wieder zurück zur Ausgangsposition. Dabei spielt es für die Aktion keine Rolle, ob die Taste mehrmals oder nur einmal gedrückt wird. Auch ein Halten der Taste hat nichts anderes als den beschriebenen Bewegungsablauf zur Folge.  Diese fliessende Bewegung, wird vom System selber durchgeführt, ohne Interaktionsmöglichkeit des Spielers während der Bewegung selbst. |
| FR4.2 | **Gegenstände in Inventar aufnehmen:** Die Interaktion durch die Spielfigur mit vordefinierten Gegenständen ist eine zentrale Funktion des Spiels und für den Spielerfolg unabdingbar.  Erreicht der Spieler mit seiner Spielfigur den *Aktionsradius eines interaktiven Elements*, muss das System ein Aktionsmenu mit entsprechenden Möglichkeiten auf dem Bildschirm anzeigen. Aus diesen Möglichkeiten, wählt der Spieler in diesem Fall „Gegenstand ins Inventar aufnehmen“. Das System fügt den Gegenstand in das Inventar des Spielers ein und der Gegenstand selbst verschwindet von seinem ursprünglichen Platz im Level. |

|  |  |
| --- | --- |
| FR5 | Gegner |
| FR5.1 | **Robotergegner patrouilliert durch das Level:** Die gegnerische Figur  (auch: „Robotergegner“ oder einfach nur „Gegner“) erhöht die Komplexität des Spielgeschehens und erschwert es dem Spieler, die Rätsel im Level zu lösen. Während dem ganzen Spiel muss sich der Robotergegner auf vordefinierten Pfaden fortbewegen. Er patrouilliert dabei am Boden und hält Ausschau nach dem Spieler. Die Bewegungen des Gegners und dessen Aktionsradius muss so modelliert sein, dass dem Spieler eine faire Chance bleibt, mit seiner Spielfigur dem Gegner auszuweichen oder sich vor ihm verstecken zu können. Dieser Gegner kann vom Spieler nicht kontrolliert oder zerstört werden. |
| FR5.2 | **Der Robotergegner löst einen Alarm aus, sobald er den Spieler entdeckt:** Sobald die Spielfigur in den Sichtradius des Robotergegners gerät, löst dieser einen akustischen Alarm aus, welcher die bis dahin gespielte Spielmusik ersetzt. Der Spieler weiss so genau, wann er entdeckt wurde. Der Alarm-Ton verstummt und wird wieder durch die Spielmusik ersetzt, sobald die Spielfigur für 2 Sekunden aus dem Sichtfeld des Roboters verschwindet. |

## Qualitätsanforderungen (Nichtfunktionale Anforderungen)

**Beschreiben sie mind. 2 nicht-funktionale Anforderungen bei ihrer Themenstellung.**Hier auch Forderungen an das Vorgehensmodell aufschreiben, was gefordert wird. Auch Anforderungen bezüglich Entscheidungen und wer was einbringt und so kann man hier reinnehmen

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nr.** | **Kurzbezeichnung** | **Status** | **P** | **V** | **K** | **R** | **Quelle** | **Datum** |
| NF1 | Konzeptionelles |  | | | | | | |
| NF  1.1 | Basis für Weiterentwicklungen | in Arbeit | 3 | 2 | 3 | high | JE | 17.09.2014 |
| NF  2.1 | Beschreibungen, Fundorte und Einsatzmöglichkeiten zu allen Gegenständen sind schriftlich festzuhalten  (Prozessanforderung) | in Arbeit | 2 | 3 | 1 | med | Team | 01.10.2014 |
| NF2 | Rechtliches |  | | | | | | |
| NF  2.1 | Urheberrechte sind bei verwendeten *Assets*  zu berücksichtigen | in Arbeit | 3 | 1 | 2 | high | Team | 01.10.2014 |

### Detailbeschreibung der nicht-funktionalen Anforderungen

|  |  |
| --- | --- |
|  | **Kurzbezeichnung** |
| NF1 | Konzeptionelles |
| NF  1.1 | **Basis für Weiterentwicklungen:** Das Basislevel, wie auch die Charaktere sollen so gestaltet sein, dass ein erfahrener Programmierer mit Praxiserfahrung in Unity innerhalb eines Arbeitstages zusätzliche Levels implementieren kann. Um die Einarbeitung in die Programmlogik zu erleichtern, muss der Programmcode sauber formatiert und kommentiert sein. |
| NF  1.2 | **Beschreibungen der Gegenstände:** Gegenstände die vom Spieler ins Inventar aufgenommen werden können, sind in einem Kapitel der Projektdokumentation zu beschreiben. Ihr Zweck, der Fundort und die Einsatzmöglichkeiten des Gegenstandes sind festzuhalten. Innerhalb des Teams einigte man sich am 29.10 auf die Darstellung in einer Matrix. |
| NF2 | Rechtliches |
| NF  2.1 | **Urheberrechte sind zu berücksichtigen:** Vor der Verwendung von *Assets*, die von Dritten im Internet angeboten werden, muss geklärt werden, in welchem rechtlichen Rahmen diese Komponenten genutzt werden dürfen. Eingekaufte, kommerzielle Assets sind als solche in der Projektdokumentation zu deklarieren. Auch ist darauf zu achten, dass keine Assets verwendet werden, die Urheberrechte von Marken verletzen, oder durch ihren Inhalt gegen andere Gesetze verstossen |

# Chancen und Risiken des Projekts

Die Entwicklungsumgebung Unity3D, die an JavaScript angelehnte Unityscript Sprache, so wie die Entwicklung geeigneter Automaten sind für die Gruppe bisher unbekannte Werkzeuge der Spieleentwicklung. Anfänglich unbekannte Themengebiete stellen von Natur aus eine grosse Herausforderung an ein realistisches Zeitmanagement.

Nachdem der Projekt Scope geklärt wurde und in der Design-Thinking-Phase erste Protoypen entwickelt wurden, konnte die Einschätzung der zeitlichen Ressourcen neu bewertet werden. Eine erste Spieleidee, verschiedene Use Cases und die Modellierung des Levels konnten das Risiko einer Fehlkalkulation minimieren. Trotzdem sind noch immer Unsicherheiten bezüglich der Machbarkeit - insbesondere was das Umsetzen der Automatentheorie betrifft - vorhanden.

Ein regelmässiger Austausch mit dem Stakeholder garantiert ein Verständnis für die Prioritäten im Projekt und mögliche Risiken können mit diesen besprochen und neu bewertet werden. Da in diesem Projekt Projektleitung und Auftraggeber durch den Dozenten verkörpert werden, wirkt sich ein Ausfall von Herr Eckerle wegen Krankheit oder andere Ereignissen stark auf unser Projekt aus.

Im Requirement Engineering besteht zudem das Risiko, nicht alle Wünsche und Anforderungen des Stakeholders erfüllen zu können. Deshalb wird diesem Teil in der ersten Phase des Projekts genug Ressourcen zugeteilt. Dennoch bietet sich auf Grund unserer gewählten agilen Entwicklungsmethode die Möglichkeit dynamisch auf Änderungswünsche und Problem-stellungen einzugehen resp. diese zu lösen.

# Rollenkonzept

| Rollenname | Funktion |
| --- | --- |
| Spieler | Die Person, welche die Spielfigur im Spiel steuert |
| Spielfigur | Die virtuelle Figur, die vom Spieler innerhalb des Systems gesteuert wird |
| Gegner | Nicht spielbare Figur im Spiel, die den Spieler am erfolgreichen Abschluss des Levels hindern soll. Im Basislevel modelliert als Roboter. |
| Entwickler | Das Projektteam bestehend aus M. Käser, F. Schwab und M. Tschanz |

# Glossar

Verantwortlich für das Glossar ist Fabian Schwab.

## Erklärungen und Übersetzungen

| Wort / Abkürzung | Bedeutung |
| --- | --- |
| HZ1  Alpha-Version | Entwicklungsstand einer Software, die schon die grundlegenden Funktionen enthält. Bereit zum Testen, aber auf keinen Fall auslieferbar. |
| HZ1  Basislevels | Als Basislevel wird, der erste Level des Spiels Rocket bezeichnet. Dieser Level 1 stellt als solcher auch das spielbare Endprodukt des Projektes dar. |
| FR3.3  Pausenmenu | Das Pausenmenu steht dem Spieler zu jedem Zeitpunkt im laufenden Spiel zur Verfügung. Der Spieler erreicht es durch Drücken der Taste ESC. Das Drücken der Taste ESC veranlasst das System dazu, das Spiel zu pausieren und das Pausenmenu mit den Optionen „Resume“, „Save game“, „Load game“ und „Quit Game“ und „Close Menu“ darzustellen. |
| FR4.2 Aktionsradius eines interaktiven Elements | Gegenstände, mit denen der Spieler interagieren kann, werden im Spiel durch eine farbige Kennung als solche ausgewiesen. Nähert sich die Spielfigur dem Gegenstand, so werden dem Spieler entsprechende Aktionsmöglichkeiten zur Auswahl gestellt, wie mit diesem Item umgegangen werden soll. |
| NF2.1  Assets | Als Assets werden Spielkomponenten bezeichnet, welche in anderen Programmen (bsp. 3D Editoren) erstellt wurden und in Unity importiert werden können. Dies können einfache Texturen sein, oder aber komplexe Modelle von Gegenständen und Figuren. |

## GUI



## Synonyme

| Wort / Abkürzung | Bedeutung |
| --- | --- |
| Rocket, System, Spiel | Als Synonyme verwendet |
| Gruppe, Team, | Als Synonyme verwendet |

# Referenzen

| ID | Titel | Autor | Format |
| --- | --- | --- | --- |
| 1 | Artificial Intelligence for Games,  Second Edition | Ian Millington  John Funge | E-Book |
| 2 | Programming Game A.I. by Example | Mat Buckland | E-Book |