|  |
| --- |
|  |
| Anforderungsspezifikation  Arcade Space Shooter  **Ueli Bühler**  **Marco Müller**  **Kevin Riesen**  17/01/2020 |
| **Berner Fachhochschule**  Departement für Technik und Informatik |

Inhaltsverzeichnis

1 Zweck des Dokuments 3

2 Vision 3

3 Projektzielsetzung 3

4 Systemabgrenzung 5

5 Anforderungen 6

6 Glossar 22

7 Literaturverzeichnis 22

8 Anhang 22

9 Versionskontrolle 23

1. Zweck des Dokuments

Dieses Dokument beschreibt die Ziele und Anforderungen für das Projekt Arcade Space Shooter.

Das Projekt wird im Rahmen des Moduls «BTI7082 Projekt 1» im Herbstsemester 2019/2020 an der Berner Fachhochschule durchgeführt und dabei fachlich und methodisch von Peter Lange begleitet.

1. Vision

Das Projekt besteht aus der Umsetzung eines Videospiels, dessen Inhalt es ist, mit einem Raumschiff umherzufliegen, feindliche Raumschiffe zu bekämpfen und ein definiertes Ziel zu erreichen.

Ziel ist es, ein unterhaltsames Spiel zu entwickeln, welches durch Arcade-Spielelemente wie die Oben-herab-Sicht oder einfache Steuerung Erinnerungen an ältere Videospiele weckt.

Der Spieler des Spiels soll sich mit unserer Applikation in die Vergangenheit zurückversetzt fühlen und dabei den Spielspass geniessen können.

1. Projektzielsetzung
   1. Ausgangslage

Früher war alles besser. Wer hat diesen Spruch nicht schon gehört und sich dabei so seine Gedanken gemacht? Gemäss diesem Ansatz wird in diesem Projekt ein Space Shooter nach Arcade Vorbild erstellt. Zum Arcade-Genre gehören Spiele, welche sich durch einfache Spielmechaniken und eine intuitive Steuerung auszeichnen. Massgebend für dieses Genre sind die heute nicht mehr häufig anzutreffenden Arcade-Automaten. Das Spiel siedelt sich in diesem Umfeld an um eine Lücke zu füllen. Arcadespiele sind nicht mehr so verbreitet und wir möchten mit unserem Projekt diesem unterhaltsamen Genre wieder zu mehr Bekanntheit verhelfen. Damit sich nicht jeder Spieler einen Spielautomaten für zuhause beschaffen muss, wird das Spiel auf dem Computer gespielt werden können.

Das Projekt startet ohne bestehende Applikation, womit es sich um eine Neuentwicklung handelt.

* 1. Stakeholder

Die relevanten Stakeholder sind:

* Peter Lange – Betreuer
* BFH – begutachtende Institution
* K. Riesen, M. Müller, U. Bühler – Durchführung und Entwicklung
* Personen, welche das Spiel spielen werden
  1. Projektziele
     1. Ziele der Stakeholder
* Personen, welche das Spiel spielen werden
  + Spielen eines lauffähigen Spieles.
  + Ein Arcadespiel wie früher an einem Spielautomaten spielen.
  + Der Spieler erfreut sich an einem unterhaltsamen Spiel.
* Entwickler
  + Durchführung eines erfolgreichen Projektes.
  + Entwicklung eines lauffähigen Spieles.
  + Erfahrungen in der Spieleentwicklung sammeln und anwenden.
  + Sich mit den Herausforderungen moderner Spieleentwicklungs-Engines auseinandersetzen.
* Betreuer / BFH
  + Betreuung, Unterstützung und Abnahme einer erfolgreichen Projektdurchführung.
    1. Priorisierung

Da es sich bei einem Computerspiel um eine aufwendige und komplexe Applikation handelt, werden die Prioritäten wie folgt gesetzt. Das Projektvorgehen wird iterativ sein, somit werden Ziele erster Priorität zuerst abgeschlossen, bevor ein neues Ziel angefangen wird.

1. **Technische Umsetzung**  
   Der Fokus des Projekts liegt auf dem Programmieren des Spiels. Bevor neue Elemente hinzugefügt werden, muss die technische Umsetzung der zugrundeliegenden Ziele erfolgt sein.
2. **Gameplay**  
   Neue Elemente die das Spielerlebnis verbessern, werden hinzugefügt, ohne ein kompliziertes Design vorauszusetzen.
3. **Design**  
   Die Elemente des Spiels werden graphisch aufgewertet um dem Spieler ein besseres Erlebnis zu bieten.
4. Systemabgrenzung
   1. Prozessumfeld

Als Spiel hat das Resultat des Projekts keine Geschäftsprozesse zu unterstützen. Das Prozessumfeld wird somit in dieser Anforderungsspezifikation nicht weiter erläutert.

* 1. Systemumfeld

In diesem Projekt möchten wir ein möglichst einfaches Spiel zur Verfügung stellen. Dazu muss ein potentieller Spieler das Spiel nur installieren und kann mit wenigen Klicks gleich das Level starten und mit dem Spiel beginnen.

In diesem Projekt existieren nur Spieler und System.

Das Spiel ist dabei in sich selbst gekapselt und benötigt deshalb keine Kommunikation mit anderen Systemen. Das Spiel läuft lokal auf einem Computer und benötigt keine Verbindung zum Internet.



* 1. Nicht unterstützte Projektziele
* Das Spiel soll keine Mehrspielerfunktionalitäten besitzen.
* Das Spiel muss keine Vielfältigkeit in der Levelauswahl aufweisen. Die Grundanforderung ist, ein spielbares Level zu haben. Eine allfällige Erweiterung durch einen dynamisch generierten Level wäre möglich.
* Das Projekt verfolgt keine kommerziellen Zwecke und wird daher nicht veröffentlicht oder vermarktet.

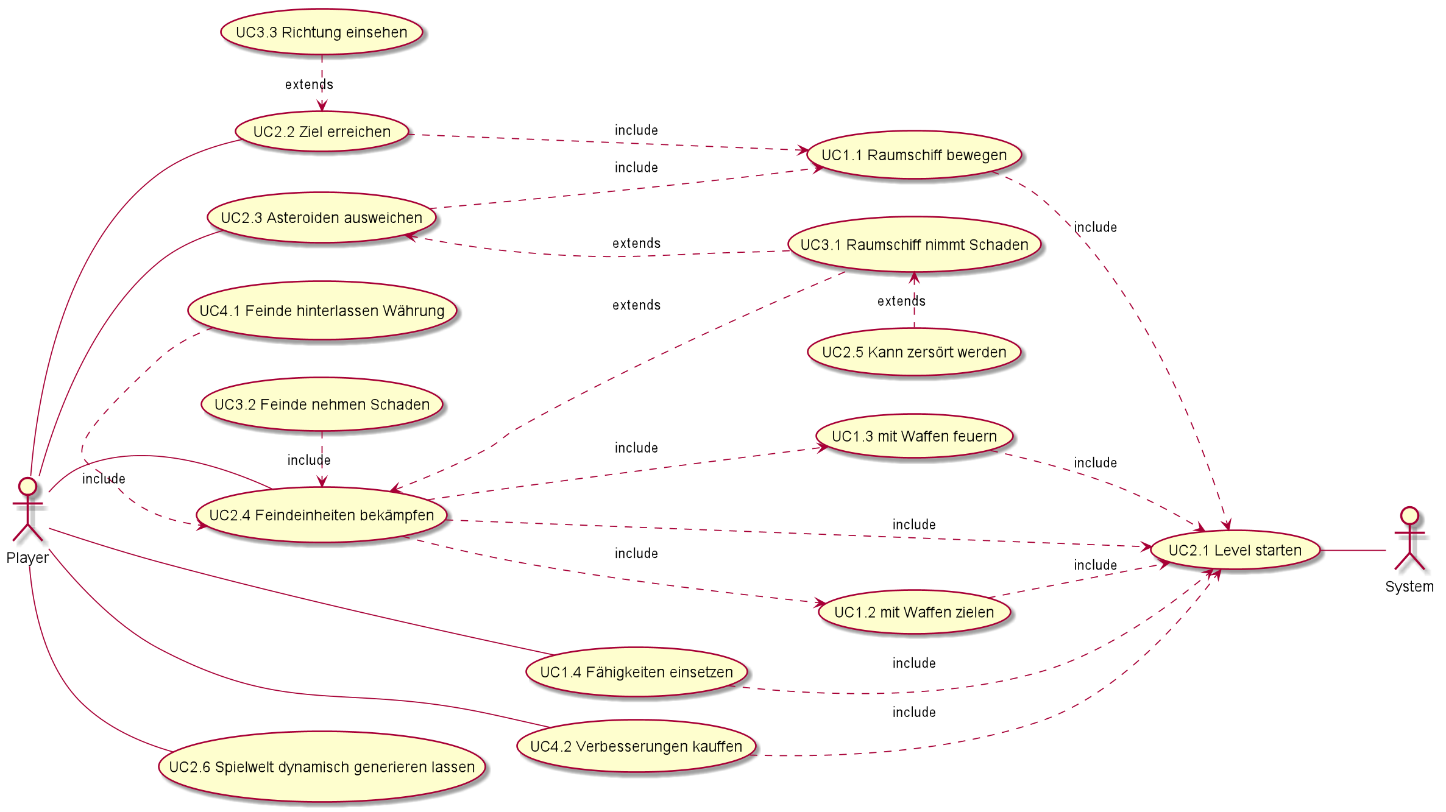
1. Anforderungen
   1. Quellen und Vorgehen

Als Ausgangslage für die Use Cases werden wir uns an bestehenden Spielen und deren Mechaniken orientieren. Bestehende Arcadespiele dienen hierbei als Hauptinspirationsquelle, zu welchen wir von Youtube und Wikipedia unsere Informationen beziehen.

Zusätzlich dient uns die Homepage von Unity (unity.com) als Informationsquelle, was mit der Engine alles realisierbar ist.

Mit den daraus gewonnen Informationen werden wir im Team Brainstorming betreiben um für uns wichtige und relevant erscheinende Anforderungen an das Projekt herauszukristallisieren. Was für dieses Projekt relevant ist, orientiert sich an der Priorisierung in Kapitel 3.3.2.

* 1. Funktionale Anforderungen
     1. Use Cases



Priorität: M = Muss / O = Optional / W = Wunsch

* + - 1. Der Spieler kann das Raumschiff kontrollieren.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **UC1.1** | Freigegeben | M | Der Spieler kann das Raumschiff mit Tastaturinput bewegen. |
| **UC1.2** | Freigegeben | M | Der Spieler kann per Mausbewegung mit den Waffen zielen. |
| **UC1.3** | Freigegeben | M | Der Spieler kann per Mausklick mit den Waffen feuern. |
| **UC1.4** | Freigegeben | W | Der Spieler kann per Nummerntasten Fähigkeiten Einsetzen |

* + - 1. Der Spieler kann einen Level spielen.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **UC2.1** | Freigegeben | M | Der Spieler kann ein Level starten |
| **UC2.2** | Freigegeben | M | Der Spieler kann in der Spielwelt zu einem Ziel gelangen, um das Spiel zu gewinnen. |
| **UC2.3** | Freigegeben | M | Der Spieler kann Hindernissen wie Asteroiden ausweichen. |
| **UC2.4** | Freigegeben | M | Der Spieler kann Feindeinheiten bekämpfen. |
| **UC2.5** | Freigegeben | M | Der Spieler kann durch Schaden am Raumschiff zerstört werden. |
| **UC2.6** | Freigegeben | O | Der Spieler kann eine Spielewelt dynamisch generieren. |

* + - 1. Der Spieler kann Informationen über das Spiel dem Userinterface entnehmen.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **UC3.1** | Freigegeben | M | Der Spieler kann seine aktuellen Lebenspunktveränderungen einsehen. |
| **UC3.2** | Freigegeben | O | Der Spieler kann die aktuellen Lebenspunktveränderungen der Feinde einsehen. |
| **UC3.3** | Freigegeben | O | Der Spieler kann die Richtung und Distanz zum Ziel einsehen. |

* + - 1. Der Spieler kann das Spielerschiff mit Upgrades ausstatten.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **UC4.1** | Freigegeben | W | Der Spieler erhält für besiegte Gegner eine Menge einer Währung. |
| **UC4.2** | Freigegeben | W | Der Spieler kann im Austausch einer Währung Verbesserungen für sein Schiff beziehen. |

* + 1. Detaillierte Anforderungen
       1. Der Spieler kann das Raumschiff mit Tastaturinput bewegen

|  |  |
| --- | --- |
| **Name** | UC1.1 |
| **Akteure** | Spieler und System |
| **Trigger** | Der Spieler befindet sich im Level und drückt eine zur Steuerung definierte Taste auf der Tastatur. |
| **Essenzielle Schritte** | 1. Der Spieler betätigt eine Steuertaste zum Steuern des Raumschiffes. 2. Das Raumschiff des Spielers wird entsprechend des Inputs bewegt. |
| **Erweiterungen** | 2a. Vorwärtstaste («W»)  2a1. Das Raumschiff bewegt sich vorwärts  2b. Rückwärtstaste («S»)  2b1. Das Raumschiff bewegt sich rückwärts  2c. Richtungstaste rechts («D»)  2c1. Das Raumschiff dreht sich nach rechts  2d. Richtungstaste links («A»)  2d1. Das Raumschiff dreht sich nach links |
| **Fehlersituationen** | Das Universum ist begrenzt. Fliegt der Spieler gegen die Begrenzung nimmt er Schaden. Die Begrenzung wird dazu sichtbar gemacht. Dies wird anhand eines Asteroidengürtels passieren. |
|  |  |

* + - 1. Der Spieler kann per Mausbewegung mit den Waffen zielen

|  |  |
| --- | --- |
| **Name** | UC1.2 |
| **Akteure** | Spieler und System |
| **Trigger** | Der Spieler befindet sich im Level und bewegt die Maus. |
| **Essenzielle Schritte** | 1. Der Spieler bewegt die Maus zum Zielen. 2. Ein Symbol zur Darstellung der Zielrichtung erschein an der Position der Maus. 3. Die Waffen des Spielerschiffes richten sich in die Richtung, in die gezielt wird. |
| **Erweiterungen** | - |
|  |  |

* + - 1. Der Spieler kann per Mausklick mit den Waffen feuern

|  |  |
| --- | --- |
| **Name** | UC1.3 |
| **Akteure** | Spieler und System |
| **Trigger** | Der Spieler befindet sich im Level und klickt mit der Maus. UC 1.2 wird erfüllt. |
| **Essenzielle Schritte** | 1. Der Spieler klickt mit der Maus um zu feuern. 2. Die Waffen des Spielerschiffes feuern Geschosse in die Richtung, in die gezielt wurde. 3. Wird ein Feind getroffen, wird diesem Schaden zugefügt. |
| **Erweiterungen** | 3a. Der zugefügte Schaden übersteigt die verbleibenden Lebenspunkte des Feindes  3a1. Der Feind wird zerstört.  3b. Das Geschoss trifft auf ein Hindernis  3b1. Das Geschoss verschwindet.  3c. Das Geschoss trifft auf kein Objekt.  3c1. Das Geschoss fliegt weiter bis es auf ein Hindernis trifft oder eine fixe Zeitspanne vergangen ist und verschwindet anschliessend. |
|  |  |

* + - 1. Der Spieler kann per Nummerntasten Fähigkeiten Einsetzen

| **Name** | UC1.4 |
| --- | --- |
| **Akteure** | Spieler und System |
| **Trigger** | Der Spieler befindet sich im Level und drückt eine für Fähigkeiten definierte Taste auf der Tastatur. |
| **Essenzielle Schritte** | 1. Der Spieler drückt eine Fähigkeitstaste auf der Tastatur, um diese einzusetzen. 2. Die Fähigkeit tritt in Kraft. 3. Eine Zeitspanne wird gesetzt, bis die Fähigkeit erneut eingesetzt werden kann (Cooldown genannt). 4. Die verbleibende Cooldownzeit wird im UI dargestellt. |
| **Erweiterungen** | 2a. Die Zeit für den Cooldown ist noch nicht vergangen.  2a1. Der weitere Vorgang wird abgebrochen. |
|  |  |

* + - 1. Der Spieler kann einen Level starten

|  |  |
| --- | --- |
| **Name** | UC2.1 |
| **Akteure** | Spieler, System |
| **Trigger** | Der Spieler startet das Spiel auf dem Computer. |
| **Essenzielle Schritte** | 1. Das System lädt das Hauptmenu. 2. Der Spieler drückt den «Level Starten» Knopf im Hauptmenu. 3. Das System lädt das Level. |
| **Erweiterungen** | - |

* + - 1. Der Spieler kann in der Spielwelt zu einem Ziel gelangen, um das Spiel zu gewinnen

|  |  |
| --- | --- |
| **Name** | UC2.2 |
| **Akteure** | Spieler, System |
| **Trigger** | Der Spieler berührt den Endpunkt des Levels |
| **Essenzielle Schritte** | 1. Der Spieler bewegt sich auf den Endpunkt 2. Das System beendet das Level |
| **Erweiterungen** | - |
|  |  |

* + - 1. Der Spieler kann Hindernissen wie Asteroiden ausweichen

|  |  |
| --- | --- |
| **Name** | UC2.3 |
| **Akteure** | Spieler, System |
| **Trigger** | Asteroiden erscheinen im sichtbaren Spielraum |
| **Essenzielle Schritte** | 1. Das System lässt Asteroiden auf dem Spielraum herumfliegen 2. Der Spieler nutzt die Steuerungstasten um den Asteroiden auszuweichen |
| **Erweiterungen** | 2.2a. Der Spieler fliegt in einen Asteroiden hinein.  2.2a1. Der Spieler nimmt Schaden. |
|  |  |

* + - 1. Der Spieler kann Feindeinheiten bekämpfen

|  |  |
| --- | --- |
| **Name** | UC2.4 |
| **Akteure** | Spieler, System |
| **Trigger** | Feinde erscheinen im sichtbaren Spielraum |
| **Essenzielle Schritte** | 1. Das System lässt Feinde auf dem Spielraum herumfliegen 2. Der Spieler schiesst auf die sichtbaren Feinde |
| **Erweiterungen** | 2a. Der Spieler wird von Feinden beschossen  2a1. Der Spieler nimmt Schaden, falls er getroffen wird. |
| **Fehlersituationen** | Das Universum ist begrenzt. Fliegt eine Feindeinheit in die Begrenzung wird sie zerstört. |
|  |  |

* + - 1. Der Spieler kann durch Schaden am Raumschiff zerstört werden

|  |  |
| --- | --- |
| **Name** | UC2.5 |
| **Akteure** | System |
| **Trigger** | Der Spieler wird auf 0 Lebenspunkte reduziert |
| **Essenzielle Schritte** | 1. Der Spieler wird auf 0 Lebenspunkte reduziert 2. Der Spieler wird ausgelöscht 3. Das System zeigt «Game Over» an und beendet das Level |
| **Erweiterungen** |  |
|  |  |

* + - 1. Der Spieler kann eine Spielewelt dynamisch generieren lassen

|  |  |
| --- | --- |
| **Name** | UC2.6 |
| **Akteure** | Spieler und System |
| **Trigger** | Der Spieler startet den Level. |
| **Essenzielle Schritte** | 1. Der Spieler startet den Level um das Spiel zu beginnen. 2. Das System kreiert einen temporären Level. Der temporäre Level ist ein neuer Level, der zusätzlich zum Standardlevel gestartet werden kann. 3. Das System platziert den Spieler im Level. 4. Das System platziert ein Ziel an einem zufälligen Ort im Level. 5. Das System platziert Hindernisse an zufälligen Orten im Level. 6. Das System platziert Feindeinheiten an zufälligen Orten im Level. 7. Der Spieler erhält die Möglichkeit den temporären Level zu spielen. |
| **Erweiterungen** |  |

* + - 1. Der Spieler kann seine aktuellen Lebenspunktveränderungen einsehen

|  |  |
| --- | --- |
| **Name** | UC3.1 |
| **Akteure** | Spieler, System |
| **Trigger** | Der Spieler wird getroffen |
| **Essenzielle Schritte** | 1. Das System registriert, dass der Spieler durch den Treffer Schaden erlitten hat. 2. Der Lebensbalken wird um die Menge des Schadens vermindert. Der Schaden ist nach Art des Projektils definiert. |
| **Erweiterungen** |  |
|  |  |

* + - 1. Der Spieler kann die aktuellen Lebenspunktveränderungen der Feinde einsehen

|  |  |
| --- | --- |
| **Name** | UC3.2 |
| **Akteure** | Spieler, System |
| **Trigger** | Der Feind wird getroffen |
| **Essenzielle Schritte** | 1. Das System registriert das ein Feind durch den Treffer Schaden erlitten hat. 2. Der Lebensbalken des Feindes wird um die Menge des Schadens vermindert |
| **Erweiterungen** |  |
|  |  |

* + - 1. Der Spieler kann die Richtung und Distanz zum Ziel einsehen

|  |  |
| --- | --- |
| **Name** | UC3.3 |
| **Akteure** | Spieler, System |
| **Trigger** | Der Spieler startet das Spiel |
| **Essenzielle Schritte** | 1. Das System stellt die Richtung und einen Indikator der Distanz zum Ziel auf dem Bildschirm an. Der Indikator wird als Zahlenwert ohne Verhältnis zum metrischen System dargestellt. 2. Der Spieler kann die Richtung und einen Indikator der Distanz zum Ziel auf dem Bildschirm ablesen. |
| **Erweiterungen** |  |
|  |  |

* + - 1. Der Spieler erhält für besiegte Gegner eine Menge einer Währung

|  |  |
| --- | --- |
| **Name** | UC4.1 |
| **Akteure** | Spieler, System |
| **Trigger** | Ein Feind wurde vom Spieler zerstört |
| **Essenzielle Schritte** | 1. Das System honoriert dem Spieler eine Menge einer Währung. 2. Das System benachrichtigt den Spieler über den Erhalt der Währung |
| **Erweiterungen** |  |
|  |  |

* + - 1. Der Spieler kann im Austausch einer Währung Verbesserungen für sein Schiff beziehen

|  |  |
| --- | --- |
| **Name** | UC4.2 |
| **Akteure** | Spieler, System |
| **Trigger** | Der Spieler öffnet das Verbesserungsmenu |
| **Essenzielle Schritte** | 1. Der Spieler öffnet das Verbesserungsmenu. 2. Der Spieler wählt eine verfügbare Verbesserung aus.   Mögliche Verbesserungen sind:   * Schutzschild * Rammbock * Erhöhte Feuerrate  1. Der Spieler klickt auf den Kaufknopf 2. Das System zieht dem Spieler die nötige Menge an Währung ab. 3. Das System ändert die Eigenschaften des Spielerschiffes entsprechend dem gekauften Upgrade. |
| **Erweiterungen** | 2a. Das Upgrade ist zu teuer  2a1. Das Upgrade wird als nicht verfügbar dargestellt. |
|  |  |

* 1. Qualitätsanforderungen

Priorität: M = Muss / O = Optional / W = Wunsch  
  
Q1 Benutzererlebnis

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **Q1.1** | Freigegeben | M | Ein durchschnittlicher Spieler ist in der Lage das Spiel zu gewinnen. |
| **Q1.2** | Freigegeben | O | Die Spielereingabe muss innerhalb von 0.1 Sekunden verarbeitet werden |
| **Q1.3** | Freigegeben | M | Das Spiel stürzt höchstens 1x alle 100 Spielstunden ab |

**Q2 Technologieanforderungen**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **Q2.1** | Freigegeben | M | Das Spiel ist auf Windows 10 spielbar. |
| **Q2.2** | Freigegeben | O | Ein System mit minimal einem Intel i5 Prozessor, 4GB Ram und einer Onboard-Grafikkarte ist fähig das Spiel zu betreiben. Anforderung Q2.1 muss erfüllt sein. |

* 1. Randbedingungen

Priorität: M = Muss / O = Optional / W = Wunsch

* + 1. Technologische Randbedingungen

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **R1.1** | Freigegeben | M | Das Spiel wird mit der Engine Unity realisiert |
| **R1.2** | Freigegeben | M | Die eingesetzte Programmiersprache soll C# sein. |
| **R1.3** | Freigegeben | M | Falls Modelle für Spielelemente erstellt werde müssen, sollen diese mit dem Tool Blender erstellt werden. |

* + 1. Zeitrahmen

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **R2.1** | Freigegeben | M | Bis 20.12.19 muss eine erste lauffähige Version des Spieles bereitstehen |
| **R2.2** | Freigegeben | M | Bis 22.01.20 muss das Projekt mit allen Deliverables abgeschlossen sein |

* + 1. Entwicklungsprozess

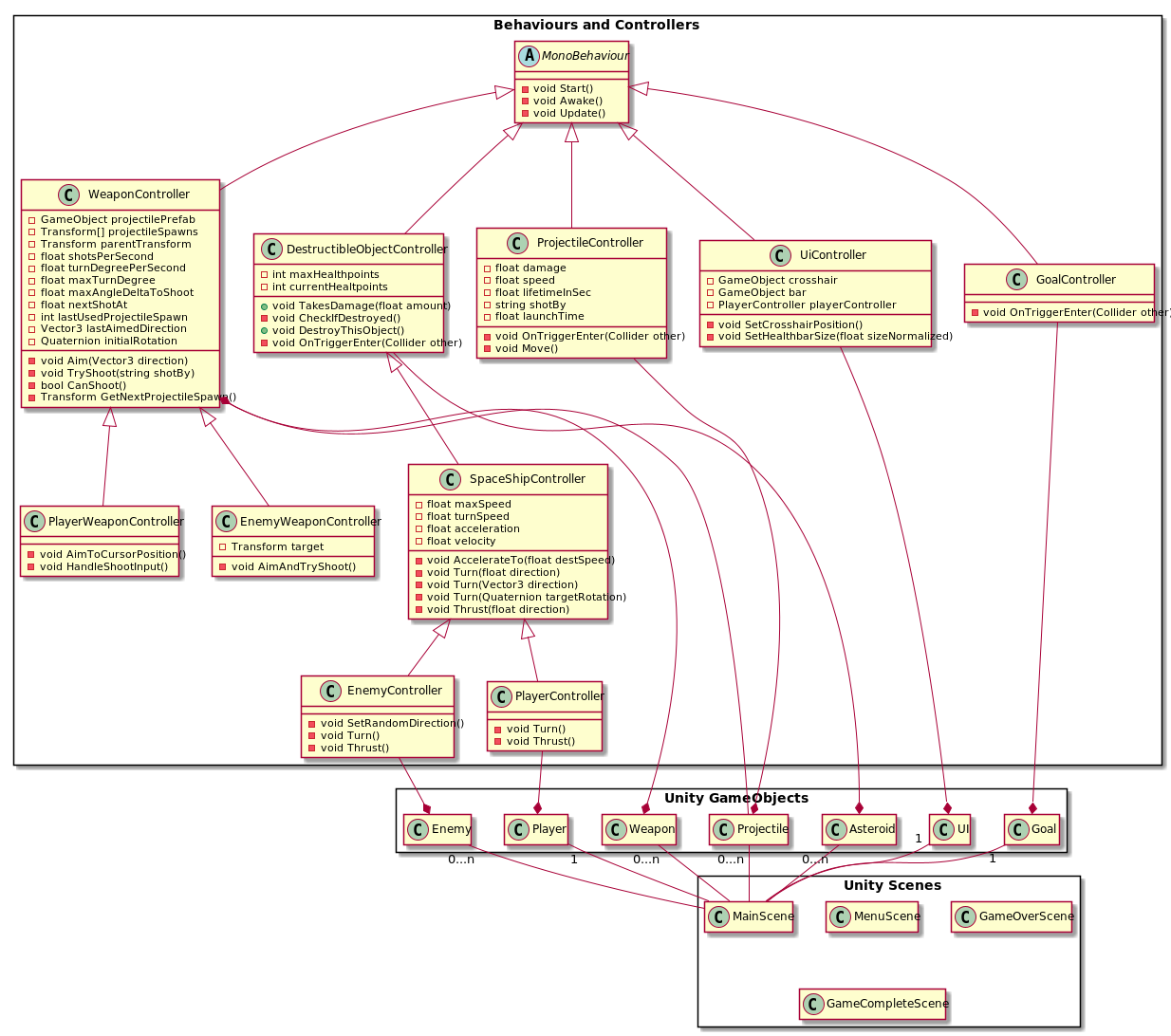
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Status** | **Prio** | **Beschreibung** |
| **R3.1** | Freigegeben | M | Das Projekt wird iterativ nach Kanban entwickelt |

* 1. Datenmodell

Die in den Use Cases beschriebenen Fälle lassen sich in folgendem Datenmodell als Objekte darstellen. Die in den Use Cases genannten Objekte werden äquivalent auf das Datenmodell übertragen.

|  |  |
| --- | --- |
| Objekt in Use Case | Objekt im Datenmodell |
| Spieler | Player |
| Feind | Enemy |
| Waffen | Weapon |
| Asteroiden | Asteroid |

Die abstrakte Klasse MonoBehaviour ist eine Basisklasse der Unity Engine. Spezifisch werden damit Verhaltensscripte definiert. In dieser Klasse werden Grundfunktionen der Engine geliefert, welche durch die Entwickler verwendet werden können.



1. Lösungsdokumentation

Im Folgenden werden die Objekte in der Hauptszene und die umgesetzten Anforderungen genauer beschrieben. Um dem Leser einen guten Überblick zu verschaffen, gehen wir hier nur auf die Kernelemente unserer Entwicklung ein.

* 1. Komponenten-diagramm der Hauptszene
  2. Umgesetzte Anforderungen

Die an dieses Projekt gestellten Muss Anforderungen konnten umgesetzt werden. Im Folgenden wird kurz beschrieben wie wir dies erreicht haben.

**UC 1: Der Spieler kann das Raumschiff kontrollieren.**

Wenn sich ein Spieler im gestarteten Level befindet ist es möglich, das Raumschiff nach den Anforderungen zu steuern. Mit der Tastatur lässt sich das Raumschiff über die Tasten W, A, S und D im zweidimensionalen Raum steuern. Der Waffenturm, welcher sich auf dem Raumschiff befindet, lässt sich über die Mausbewegung steuern. Es ist nicht möglich die Waffe 360 Grad zu drehen. Es kann nicht nach hinten geschossen werden. Mit einem Klick auf die linke Maustaste wird die Waffe betätigt. Bleibt die Maustaste gedrückt, befindet sich die Waffe im Dauerfeuermodus.

**UC2: Der Spieler kann einen Level spielen.**

Wenn ein Spieler das Spiel startet, wird als erstes ein Menu angezeigt. Über das Menu kann das Level gestartet werden. Ein Level besteht aus einer vordefinierten Anzahl Feinden und Asteroiden. Diese werden innerhalb verteilten Spawnpunkten zufällig generiert. Der Spieler kann nun mithilfe der implementierten Steuerung den Hindernissen ausweichen und zum Ziel, einer Raumstation, gelangen. Mit der Waffe können Feinde bekämpft und zerstört werden. Wird das Ziel erreicht, ist das Level gewonnen. Wenn der Spieler in einen Asteroiden fliegt oder den Spielbereich, begrenzt durch einen Asteroidengürtel, verlässt, nimmt er Schaden. Treffer durch Feindeinheiten generieren ebenfalls Schaden. Falls der Spieler keine Lebenspunkte mehr hat, wird das Raumschiff zerstört und das Level ist verloren.

**UC3: Der Spieler kann Informationen über das Spiel dem Userinterface entnehmen.**

Im Userinterface werden dem Spieler seine Lebenspunkte in Form eines Lebensbalkens dargestellt. Wird der Spieler getroffen oder kollidiert er mit Asteroiden, verliert der Spieler Lebenspunkte. Dies wird dem Spieler durch einen schrumpfenden Lebensbalken angezeigt.

1. Glossar

|  |  |
| --- | --- |
| Bezeichnung | Beschreibung |
| Arcade-Spiel | Arcade-Spiel ist eine Bezeichnung für Videospiele, welche ab den 1970er Jahren in Spielhallen, kostenpflichtig angeboten wurden. |
| Space Shooter | Ein Space Shooter ist ein Spiel, dessen Handlung im Weltall spielt. Gespielt wird häufig ein Raumschiff, mit welchem Aufgaben erfüllt werden müssen. |
| Unity | Unity ist eine Laufzeit- und Entwicklungsumgebung für Spiele. |
| Cooldown | Cooldown beschreibt das Abkühlen oder Auslaufen. Im Kontext des Spaceshooters ist damit die Zeit gemeint, bis eine Waffe nach dem Gebrauch wieder einsatzfähig ist. |

1. Literaturverzeichnis

**Unity Forum**

<https://forum.unity.com/> Stand 30.11.19

**Youtube Tutorial**

<https://www.youtube.com/watch?v=AacN8q57xJg&list=PL_eGgISVYZkfZRSzSDyG3tHNf5_DF-o5H&index=1> Stand 30.11.19

**Unity Tutorial**

<https://learn.unity.com/> Stand 30.11.19

1. Anhang
   1. Abstimmung der Anforderungen

Die Anforderungen an dieses Projekt werden in einem Meeting mit dem Stakeholder, Herr Peter Lange, abgesprochen. Dazu wird dieses Dokument jeweils aktualisiert und vor dem Meeting an Herrn Lange gesendet.

Bei der Ausarbeitung dieses Dokumentes hat sich herausgestellt, dass in den Use Cases und im Datenmodell nicht die gleiche Sprache zum beschreiben der Objekte verwendet wurde. Gelöst haben wir dies, indem wir eine Mapping-Tabelle (vgl. Kapitel 5.5) erstellt haben.

* 1. Definition of Ready – Checklist

• Backlog Items sind klein genug, um im Sprint abgeschlossen zu werden

• Backlog Items sind für jeden beteiligten Entwickler klar verständlich

• Jedes Backlog Item hat Akzeptanzkriterien (Min. 1)

• Jedes Backlog Item hat einen Wert im Sinne der Projektziele

* 1. Testkonzept

Zu jedem bearbeiteten Task wird ein jeweils separater Branch, vom Develop-Branch ausgehend, erstellt. Nach Abschluss des Tasks muss der Pullrequest von mindestens einem weiteren Entwickler abgesegnet werden, damit der Branch nach Develop zurück gemergt wird.

Punkte bei dieser Überprüfung sind folgende:

* Codereview
* Manuelles Testing der Erweiterung
* Manuelles Testing auf Erfüllung der Akzeptanzkriterien

Damit soll eine kontinuierliche Gewährleistung der bestehenden sowie neuen Funktionalitäten unserer Applikation erzielt werden.

Auf ein automatisiertest Testing wird absichtlich verzichtet, da uns aus zeitlichen Gründen die Einarbeitung in das Testingframework von Unity nicht möglich ist.

1. Versionskontrolle

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Beschreibung** | **Autor** |
| X0.1 | 23.09.2019 | Initiale Version kopiert und Angepasst | Riesen |
| 1.0 | 26.09.2019 | Vision und Zielsetzung | Bühler, Müller, Riesen |
| 1.1 | 04.10.2019 | Überarbeitung, Kapitel 4 | Müller |
| 1.1 | 04.10.2019 | Erster Entwurf Usecases | Bühler, Riesen |
| 1.2 | 09.10.2019 | Überarbeitung, Verfeinerung Usecases | Bühler, Müller, Riesen |
| 1.3 | 11.10.2019 | Überarbeitung, Kapitel 5 | Bühler, Müller, Riesen |
| 1.4 | 16.10.2019 | Überarbeitung, Erweiterung Kapitel 5 | Müller, Riesen |
| 1.5 | 18.10.2019 | Überarbeitung Datenmodel | Riesen |
| 1.5 | 18.10.2019 | Erstellung Usecasediagramm | Bühler, Riesen |
| 1.5 | 20.10.2019 | Überarbeitung Kapitel 1 - 5 | Müller |
| 1.5 | 29.10.2019 | Überarbeitung gemäss Peer Review | Müller |
| 1.5 | 30.10.2019 | Überarbeitung gemäss Peer Review | Riesen |
| 1.6 | 30.11.2019 | Überarbeitung gemäss Peer Review | Riesen, Müller, Bühler |