Design Auftrag Docker

**Inhalt**

[Projektmanagement 2](#_Toc404007453)

[Projektstrukturplan 3](#_Toc404007454)

[Softwareentwicklungsplan 4](#_Toc404007455)

[Arbeitspakete 4](#_Toc404007456)

[Stundenerfassung 7](#_Toc404007457)

[Risiken 8](#_Toc404007458)

[Besonderes 8](#_Toc404007459)

[Risikodiagramm 9](#_Toc404007460)

[Architektur 10](#_Toc404007461)

[Wesentliche Änderungen und Entscheide 10](#_Toc404007462)

[Entfernung des Renderer-Pakets 10](#_Toc404007463)

[Nutzung von Scene2D 10](#_Toc404007464)

[Game-Paket 10](#_Toc404007465)

[User-Paket 11](#_Toc404007466)

[Menus-Paket 11](#_Toc404007467)

[Paketdiagramm 11](#_Toc404007468)

[Design-Klassendiagramm 12](#_Toc404007469)

[Klassenverantwortlichkeiten 14](#_Toc404007470)

[Zusammenarbeitsdiagramme 15](#_Toc404007471)

[Systemoperation: deployContainer 15](#_Toc404007472)

[Beschreibung 15](#_Toc404007473)

[Sequenzdiagramm 15](#_Toc404007474)

[Systemoperation: calculateScore 16](#_Toc404007475)

[Beschreibung 16](#_Toc404007476)

[Sequenzdiagramm 16](#_Toc404007477)

[Systemoperation: saveGame 16](#_Toc404007478)

[Beschreibung 16](#_Toc404007479)

[Kommunikationsdiagramm 16](#_Toc404007480)

[Systemoperation: capsizeShip 17](#_Toc404007481)

[Beschreibung 17](#_Toc404007482)

[Kommunikationsdiagramm 17](#_Toc404007483)

[Systemoperation: breakShip 17](#_Toc404007484)

[Beschreibung 17](#_Toc404007485)

[Kommunikationsdiagramm 17](#_Toc404007486)

[GUI-Design 18](#_Toc404007487)

[Menüs 18](#_Toc404007488)

[Spielbildschirm 18](#_Toc404007489)

[Glossar 19](#_Toc404007490)

[Projektdomäne 19](#_Toc404007491)

[Primärbegriffe 19](#_Toc404007492)

[Sekundärbegriffe 20](#_Toc404007493)

[Projektmanagement 20](#_Toc404007494)

# Projektmanagement

Für das Projektmanagement wurde eine Liste mit allen Arbeitspaketen erstellt um die prognostizierten, sowie die aufgewendeten Stunden zu dokumentieren. Daraus wird nun auch automatisch die Stundendifferenz zwischen prognostizierten und effektiv angefallenen Aufwände für die einzelnen Phasen berechnet. Der Softwareentwicklungsplan wurde auf den neusten Stand gebracht sowie die aktuellen Iterationen der Construction Phase etwas genauer ausgeführt. Auch die Risikoeinschätzung erhielt ein kleines Update, da Christoph Mathis vom 26.11-30.11 auf Grund eines Kurses nicht an Docker arbeiten können wird.

## Projektstrukturplan

**A Management** AA Ideensuche  
 AB Spielbeschrieb  
 AC Anforderungen  
 AD Ressourcen  
 AE Projektplanung  
 AEA Risiken und Grobplanung  
 AEB Projektmanagement  
 AF Kundennutzung und Wirtschaftlichkeit

**B Entwicklungsumgebung**  
 BA Engineering und Evaluation

**C Anforderungen** CA Anwendungsfälle  
 CAA Anwendungsfalldiagramm  
 CAB System-Sequenzdiagramm  
 CB Zusätzliche Spezifikationen

**D Design**  
 DA Domänenmodell  
 DAA Domänenmodell visualisieren  
 DB Architektur  
 DBA Architektur visualisieren  
 DBB Klassenverantwortlichkeit  
 DBC Zusammenarbeitsdiagramme

**E Implementation** EA Repository  
 EAA Klassendiagramm

EB Domain  
 EBA GameObjects  
 EBAA Ship  
 EBAB Train  
 EBAC Crane  
 EBB Gamebewertung  
 EBC Gamelogik  
 EBCA InfiniteGame  
 EBCB CareerGame  
 EBCC QuickGame  
 EC User Interface  
 ECA Rendering  
 ECAA Grafiken  
 ECB Menu  
 ECC User Config & Stats  
 ECD Level  
 ED Tech. Services  
 EDA Persistence

**F Evaluation und Test**

**G Auslieferung**

## Softwareentwicklungsplan

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 23. Sep | **30. Sep** | 07. Okt | 14. Okt | **21. Okt** | 28. Okt | 04. Nov | 11. Nov | 18. Nov | 25. Nov | **02. Dez** | **09. Dez** |
|  | **Inception** | | **Elaboration** | | | **Construction** | | | | | | **Transition** |
|  | **I1** | | **E1** | | | **C1** | | **C2** | | **C3** | | **T1** |
| RH | A | | A, C, D | | | E, EBB | | D, DBC, E, EBB, ECC, ECD | | E, ECC, ECD, G | | G |
| YM | A | | A, C, D | | | E, EA, ECA | | D, DB, DBC, EAA, EBAC, EAA | | E, EBCA, ECA | |  |
| CM | A | | A, C, D | | | E, EBAA, EBC | | D, DBC, EBAA, EBC, ECC | | E, EBAA, EBCC | | G |
| EW | A | | A, C, D | | | E, EBAB, ECB | | D, DBB, DBC, EBAB, EBCC, ECA | | E, EBCB, ECC | | G |
|  |  | **M1** |  |  | **M2** |  |  |  |  |  | **M3** | **M4** |
|  |  | **P1** |  |  | **P2** |  |  |  | **P3** |  |  | **P4** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Meilensteine: Projektschiene** | | | | | **Meilensteine: Projekt Docker** | | | | |  | **Legende** |  |  |
| 30.09.2014 | Präsentation Projektskizze | | | P1 | 30.09.2014 | Inception Abschluss | | | M1 |  | **Kürzel** | **Name** | |
| 21.10.2014 | Präsentation Anforderungen | | | P2 | 21.10.2014 | Elaboration Abschluss | | | M2 |  | RH | Remo Höppli | |
| 18.11.2014 | Präsentationen Design | | | P3 | 02.12.2014 | Construction Abschluss | | | M3 |  | YM | Yacine Mekesser | |
| 09.12.2014 | Schlusspräsentationen | | | P4 | 09.12.2014 | Transition Abschluss | | | M4 |  | CM | Christoph Mathis | |
|  |  |  |  |  |  |  |  |  |  |  | EW | Emily Wangler | |

## Arbeitspakete

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Auftrag** | **Arbeitspaket** | **Kennung** | **Wer** | **Prognostiziert** | **Aufwand** | **Differenz** |
| I1 | Projekt | Ideensuche | A | Alle (\*4) | 8.0 | 8.0 | 0.0 |
| I1 | Projektskizze | Idee | AA | EW | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Hauptanwendungsfall | AB | EW | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Kundennutzung | AF | CM | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Wirtschaftlichkeit | AF | CM | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Risiken | AEA | RH | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Projektplanung | AEB | RH | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Ressourcen | AD | RH | 2.0 | 2.0 | 0.0 |
| I1 | Projektskizze | Weitere Anforderungen | AC | YM | 1.0 | 1.0 | 0.0 |
| I1 | Projektskizze | Abgrenzungen | AC | YM | 1.0 | 1.0 | 0.0 |
| I1 | Projekt | Evaluation ASDK | BA | YM | 6.0 | 6.0 | 0.0 |
| I1 | Projekt | Besprechungen | C &DB | Alle (\*4) | 16.0 | 16.0 | 0.0 |
| E1 | Analyse | Projektmanagement | AEB | RH | 4.0 | 4.0 | 0.0 |
| E1 | Analyse | Anwendungsfälle | CA | Alle (\*4) | 8.0 | 8.0 | 0.0 |
| E1 | Analyse | Anwendungsfalldiagramm | CAA | CM | 1.0 | 1.0 | 0.0 |
| E1 | Analyse | Domänenmodell | DA | RH | 2.0 | 2.0 | 0.0 |
| E1 | Analyse | Erste Architektur | DB | YM | 4.0 | 4.0 | 0.0 |
| E1 | Analyse | Zusätzliche Spezifikationen | CB | EW | 4.0 | 4.0 | 0.0 |
| E1 | Analyse | System-Sequenzdiagramm | CAB | CM | 1.0 | 1.0 | 0.0 |
| E1 | Analyse | Systemoperationen | CA | CM | 2.0 | 2.0 | 0.0 |
| E1 | Analyse | Glossar | D | YM | 2.0 | 2.0 | 0.0 |
| E1 | Projekt | Besprechungen | D | Alle (\*4) | 16.0 | 16.0 | 0.0 |
| C1 | Projekt | Besprechungen | E | Alle (\*4) | 16.0 | 16.0 | 0.0 |
| C1 | Projekt | Repository | EA | YM | 1.0 | 1.0 | 0.0 |
| C1 | Projekt | Klassendiagramm | EAA | YM | 1.0 | 1.0 | 0.0 |
| C1 | Projekt | Rendering | ECA | YM | 2.0 | 5.0 | 3.0 |
| C1 | Projekt | Grafiken | ECAA | YM | 4.0 | 4.0 | 0.0 |
| C1 | Projekt | Gamebewertung | EBB | RH | 3.0 | 4.0 | 1.0 |
| C1 | Projekt | Ship Logik | EBAA | CM | 3.0 | 3.0 | 0.0 |
| C1 | Projekt | Game Logik | EBC | CM | 5.0 | 5.0 | 0.0 |
| C1 | Projekt | Train Logik | EBAB | EW | 3.0 | 1.0 | 2.0 |
| C1 | Projekt | Menu | ECB | EW | 3.0 | 1.0 | 2.0 |
| C1 | Design | Projektmanagement | E | RH | 4.0 | 5.0 | 1.0 |
| C2 | Design | Architektur | DB | YM | 2.0 | 1.0 | 1.0 |
| C2 | Design | Projektmanagement | E | RH | 4.0 | 4.0 | 0.0 |
| C2 | Design | Klassendiagramm | EAA | YM | 1.0 | 1.0 | 0.0 |
| C2 | Design | Klassenverantwortlichkeit | DBB | EW | 2.0 | 2.0 | 0.0 |
| C2 | Design | Zusammenarbeitsdiagramme | DBC | Alle (\*4) | 8.0 | 6.0 | 2.0 |
| C2 | Design | Dokumentfinish | D | Alle (\*4) | 4.0 | 4.0 | 0.0 |
| C2 | Projekt | Gamebewertung | EBB | RH | 4.0 | 2.0 | 2.0 |
| C2 | Projekt | Ship Logik | EBAA | CM | 4.0 | 3.0 | 1.0 |
| C2 | Projekt | Train Logik | EBAB | EW | 2.0 | 0.0 | 2.0 |
| C2 | Projekt | Rendering | ECA | YM | 2.0 | 2.0 | 0.0 |
| C2 | Projekt | Game Logik | EBC | CM | 4.0 | 4.0 | 0.0 |
| C2 | Projekt | Crane Logik | EBAC | YM | 4.0 | 2.0 | 2.0 |
| C2 | Projekt | Quick Game | EBCC | EW | 4.0 | 4.0 | 0.0 |
| C2 | Projekt | Persistence | ECA | EW | 4.0 | 3.0 | 1.0 |
| C2 | Projekt | Level | ECD | RH | 2.0 | 3.0 | 1.0 |
| C2 | Projekt | Statistik | ECC | RH | 1.0 | 1.0 | 0.0 |
| C2 | Design | Präsentation Demo | D | CM | 2.0 | 2.0 | 0.0 |
| C2 | Design | Bewertung und Levelgenerator | D | RH | 3.0 | 2.0 | 1.0 |
| C2 | Design | Präsentation Grafik | D | YM | 2.0 | 2.0 | 0.0 |
| C2 | Projekt | Score Bildschirm | ECC | CM | 3.0 | 4.0 | 1.0 |
| C3 | Projekt | Quick Game | EBCC | CM | 4.0 |  | 4.0 |
| C3 | Projekt | Career Game | EBCB | EW | 6.0 |  | 6.0 |
| C3 | Projekt | Infinite Game | EBCA | YM | 6.0 |  | 6.0 |
| C3 | Projekt | Statistik Screen | ECC | EW | 3.0 |  | 3.0 |
| C3 | Projekt | Level for Career Game | ECD | RH | 4.0 |  | 4.0 |
| C3 | Projekt | Handicapmenu | ECC | RH | 2.0 |  | 2.0 |
| C3 | Projekt | Besprechungen | E | Alle (\*4) | 8.0 |  | 8.0 |
| C3 | Projekt | Rendering | ECA | YM | 2.0 |  | 2.0 |
| C3 | Schlusspräsentation | Projektmanagement | G | RH | 4.0 |  | 4.0 |
| C3 | Projekt | Ship Logik (Animation) | EBAA | CM | 4.0 |  | 4.0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Stundenerfassung |  |  |  |  |  |  |  |
| **Aufwände** | | | | | | | |
| **Name** | **I1** | **E1** | **C1** | **C2** | **C3** | **T1** | **Total** |
| Remo Höppli | 12 | 12 | 13 | 14.5 | 0 | 0 | 51.5 |
| Yacine Mekesser | 14 | 12 | 15 | 10.5 | 0 | 0 | 51.5 |
| Christoph Mathis | 10 | 10 | 12 | 15.5 | 0 | 0 | 47.5 |
| Emily Wangler | 10 | 10 | 6 | 11.5 | 0 | 0 | 37.5 |
| Total | 46 | 44 | 46 | 52 | 0 | 0 | 188 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Prognose** | | | | | | | |
| **Name** | **I1** | **E1** | **C1** | **C2** | **C3** | **T1** | **Total** |
| Remo Höppli | 12 | 12 | 11 | 17 | 12 | 0 | 64 |
| Yacine Mekesser | 14 | 12 | 12 | 14 | 10 | 0 | 62 |
| Christoph Mathis | 10 | 10 | 12 | 16 | 10 | 0 | 58 |
| Emily Wangler | 10 | 10 | 10 | 15 | 11 | 0 | 56 |
| Total | 46 | 44 | 45 | 62 | 43 | 0 | 240 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Differenz (verfügbare Stunden)** | | | | | | | |
| **Name** | **I1** | **E1** | **C1** | **C2** | **C3** | **T1** | **Total** |
| Remo Höppli | 0 | 0 | -2 | 2.5 | 12 | 0 | 12.5 |
| Yacine Mekesser | 0 | 0 | -3 | 3.5 | 10 | 0 | 10.5 |
| Christoph Mathis | 0 | 0 | 0 | 0.5 | 10 | 0 | 10.5 |
| Emily Wangler | 0 | 0 | 4 | 3.5 | 11 | 0 | 18.5 |
| Total | 0 | 0 | -1 | 10 | 43 | 0 | 52 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Gesamtprognose | 400 |  |  |  |  |  |  |
| Bisher benötigt | 188 |  |  |  |  |  |  |
| Verbleibend | 212 |  |  |  |  |  |  |

## Risiken

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Risiko** | **Beschreibung** | **EW** | **AW** | **Massnahmen** |
| 1 | ZHAW Netzwerk | ZHAW Server sind aufgrund eines Wartungsfensters oder Ausfalls nicht erreichbar. | Sehr wahr- scheinlich | Gering | Git benutzen. |
| 2 | Motivation | Motivation während des Semesters lässt nach. | Wahr-scheinlich | Mittel | Arbeiten gerecht verteilen. Teamgeist pflegen und klare gemeinsame Ziele definieren. |
| 3 | Probleme mit der Entwicklungsumgebung | Probleme mit Framework oder Android SDK. | Möglich | Hoch | Gemeinsames Einrichten der Entwicklungsumgebungen und gegenseitige Unterstützung bei Problemen |
| 4 | Hardwareausfall | Ein Handy oder Notebook fällt aus. | Möglich | Hoch | Material sorgfältig behandeln und bei einem Ausfall zeitig für Ersatz sorgen. |
| 5 | Sound & Grafik | Zeit für die Implementation wird knapp, Mittel für die Realisierung reichen nicht aus. | Möglich | Hoch | Sound weglassen und/oder Grafik vereinfachen. |
| 6 | Personaldefizit | Ausfälle durch Krankheit oder Unfall, viel zu tun bei der Arbeit. **WK Yacine 24.11-12.12! Kurs Christoph 26.11-30.11** | **Sehr wahr-scheinlich** | Mittel | Viel Wissenstransfer & flexible Planung. Verlängerung der Construction Phase, Verkürzung der Transition Phase |
| 7 | Schlechtes Zeitmanagement | Fehleinschätzung, Zeitmangel auf Grund von Teilzeit Pensum. | Unwahr- scheinlich | Hoch | Realistischen Zeitplan erstellen. Verzögerungen frühzeitig erkennen und aufholen. |
| 8 | Know-how Defizit | Das Know-how im Team oder bei einzelnen Mitgliedern führt zu Verzögerungen | Möglich | Gering | So viel Wissenstransfer betreiben wie möglich. |
| **EW: Eintrittswahrscheinlichkeit AW: Auswirkung** | | |  |  |  |

## Besonderes

Das Risiko mit der Nummer 6 wird eintreten, da Christoph Mathis vom 26.11.-30.11 einen Kurs in Strassburg besucht und deshalb nicht für Docker arbeiten kann. Yacine Mekesser wird während seines WKs die Möglichkeit bekommen für Docker zu arbeiten, was das Personaldefizit etwas abschwächt. Während der Schlusspräsentation wird Yacine Mekesser jedoch leider abwesend sein. Insgesamt sollten hier keine grösseren Probleme auftauchen, da das Projekt ziemlich gut im Zeitplan liegt. Weitere Änderungen in der Personaleinplanung werden während des Projektverlaufs aktualisiert.

## Risikodiagramm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Eintrittswahrscheinlichkeit →** | Sehr wahr- scheinlich | 1 | **6** |  |
| Wahrscheinlich |  | 2 |  |
| Möglich | 8 |  | 3, 4, 5 |
| Unwahrscheinlich |  |  | 7 |
|  |  | Niedrig | Mittel | Hoch |
|  |  | **Auswirkung →** | | |

# Architektur

Im Zuge der Designphase haben wir einen ersten Entwurf der Systemarchitektur erstellt. Nun, in der Constructionphase, hat sich dieser Entwurf konkretisiert und wurde entsprechend neuer Erkenntnisse angepasst. Im Folgenden sollen nun die wesentlichen Designentscheide und Änderungen zur letzten Version vorgestellt und begründet werden.

## Wesentliche Änderungen und Entscheide

### Entfernung des Renderer-Pakets

In der ersten Version der Architektur war geplant, die visuelle Darstellung der Spielobjekte in eigene Renderklassen in einem Renderer-Paket auszulagern. Diese Idee wurde aber bereits in der ersten Entwurfsphase wieder verworfen. Grund dafür war in erster Linie, dass sich dieses Konzept nicht gut mit dem Scene2D-System von libGDX vertrug, welches wir nutzen wollten. Dort wird davon ausgegangen, dass alle Actors (also alle Spielobjekte) selber für ihre Darstellung verantwortlich sind. Natürlich könnten unsere GameObjects in ihren Rendermethoden die Renderobjekte aufrufen, aber das würde zu einer Verletzung des Schichtenprinzips führen, weil Abhängigkeiten nur zu unteren Schichten bestehen dürfen.

Weiterhin war unsere Begründung, das Rendering auszulagern, in erster Linie diese, dass wir vorläufige Renderfunktionen später einfach mit den finalen austauschen können. In der Realität kamen wir aber mit der Grafik sehr zügig vorwärts, so dass sich uns in diesem Bezug gar kein Vorteil mehr bot.

Aus diesen Gründen entschieden wir uns, auf das Renderer-Paket komplett zu verzichten und die Renderfunktionen direkt in den GameObjects zu implementieren.

### Nutzung von Scene2D

Dieser Punkt ist nicht im Paketdiagramm ersichtlich, aber dennoch ein essentieller Entscheid. Das Java-Spieleentwicklungsframework libGDX bietet den Scene 2D Scene Graph um UI- und Spielelemente einfach verwalten zu können. Das Konzept beruht auf der Analogie mit einem Bühnenspiel: Ein Stage-Objekt dient als Container für Actor-Objekte und handelt sowohl Input-Events als auch das Rendering. Actor-Objekte können auch gruppiert und so hierarchisch strukturiert werden. Die weiteren Features von Scene2D, insbesondere auch in Bezug auf das Rendering, können unter <https://github.com/libgdx/libgdx/wiki/Scene2d> eingesehen werden.

Das System eignet sich sehr gut für unseren relativ simplen Spielaufbau und nimmt uns viel Arbeit ab. So erben alle Klassen in GameObjects (Ship, Train, Container, Crane) von der Actor-Klasse und werden vom Game in eine Stage eingefügt.

### Game-Paket

Die Levelklasse wurde zumindest vorläufig in das Game-Paket verschoben. Fachlich macht das Sinn, weil ausschliesslich das CareerGame die Levelklasse nutzt und der Level schliesslich ein Spiel definiert. Sollte es sich ergeben, dass wir noch weitere Levelklassen benötigen, wäre es dann sinnvoll, ein :Domain::Level oder :Domain::Game::Level Paket zu erstellen.

Ausserdem wurde noch die neue Klasse LoadRating ins Game-Paket aufgenommen. Sie enthält die Bewertungsalgorithmen, die von den Games genutzt werden.

### User-Paket

Im User-Paket wurde um die Handicap-Klasse erweitert. Diese enthält und verwaltet die Handicap-Einstellungen des Benutzers (vgl. Domänenmodell).

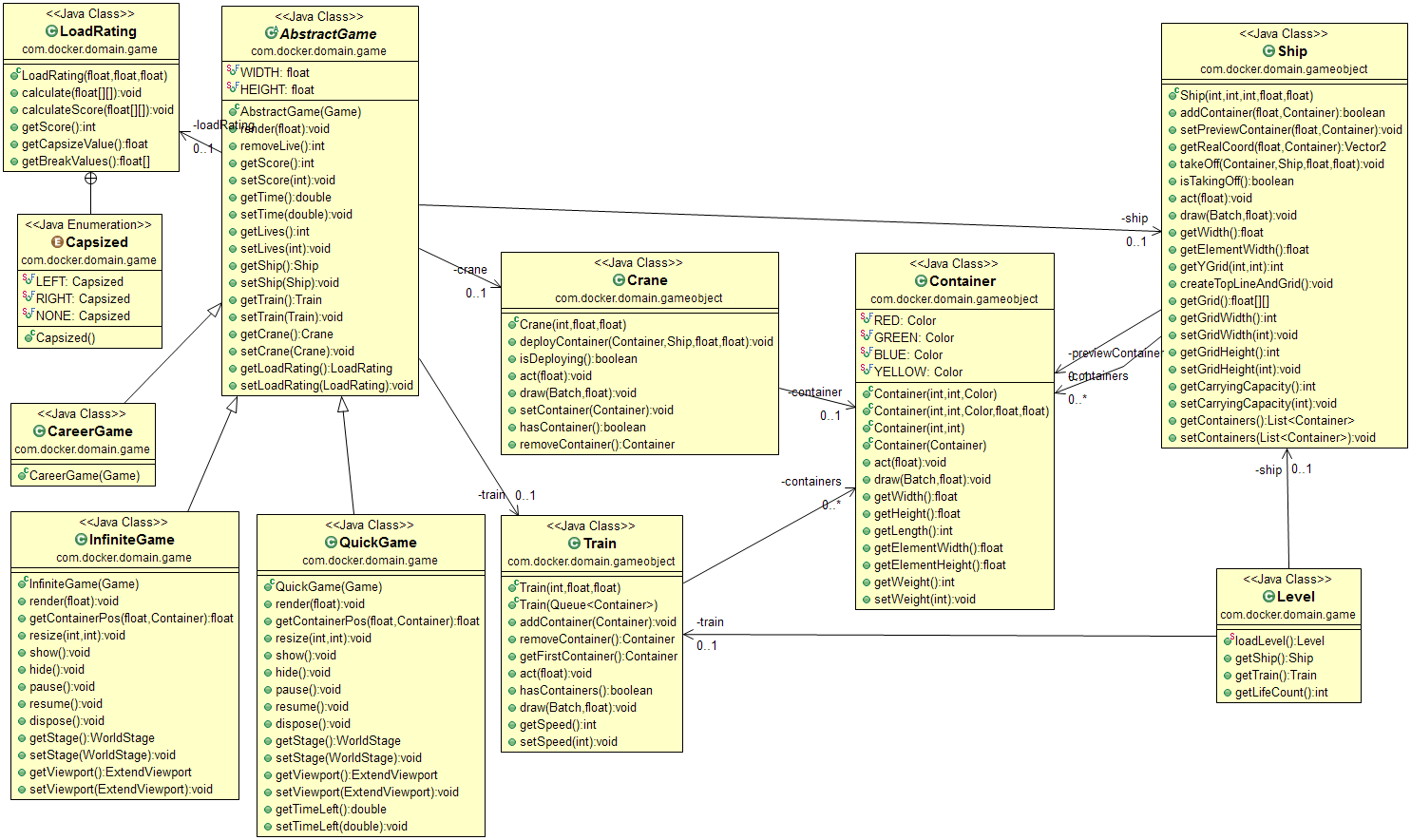
### Menus-Paket

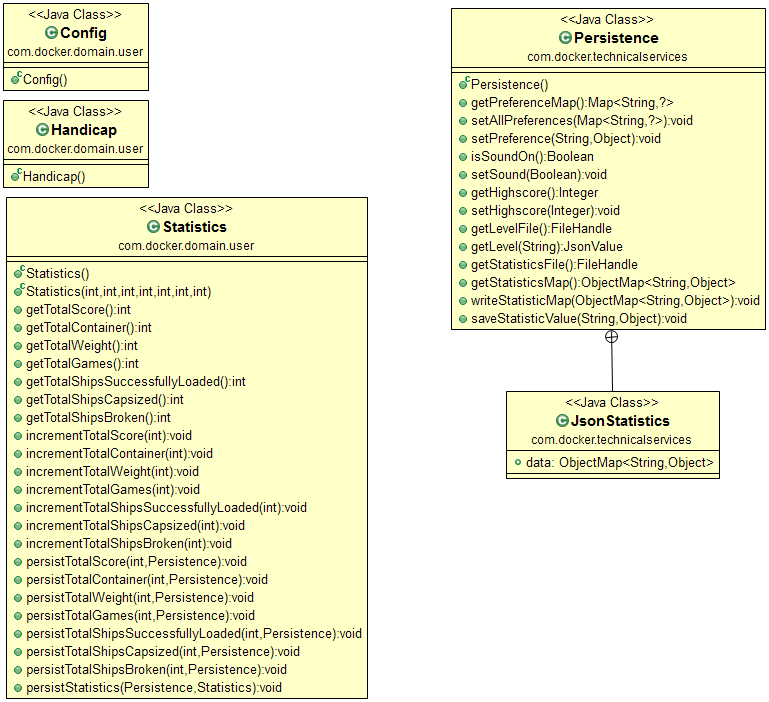
Hier kam neu die Klasse StatisticsMenu hinzu. Dem Namen entsprechend handelt es sich um den Statistikbildschirm im Menü.

## Paketdiagramm



# Design-Klassendiagramm





# Klassenverantwortlichkeiten

|  |  |
| --- | --- |
| Klasse | Verantwortlichkeiten |
| Statistics | Speichert statistische Daten zur Benutzung vom Spiel. |
| Persistence | Bietet Funktionalitäten um Daten auf dem Gerät zu speichern und vom Gerät zu lesen. |
| Config | Beinhaltet die Verwaltung der Applikationseinstellungen. |
| Handicap | Beinhaltet die Verwaltung der Spieleinstellungen. |
| Ship | Verwaltet die Container und welche Positionen noch frei sind. |
| Container | Stellt einen Container dar und hat eine Länge sowie ein Gewicht. |
| Level | Enthält die Logik ein Level zu generieren oder zu laden. |
| Train | Bringt die Container für das Beladen des Schiffes. |
| Crane | Platziert Container auf dem Schiff. |
| AbstractGame | Enthält die übergreifende Spiellogik, welche bei allen Spielmodi dieselbe ist. |
| LoadRating | Berechnet anhand einer zweidimensionalen Matrix den Kippwert, die Bruchwerte der einzelnen Abschnitte und den Punktestand. |
| CareerGame | Lädt ein vordefiniertes Level und steuert alle Spielevents. |
| Quickgame | Lädt ein zufällig generiertes Level. Steuert alle Spielevents. |
| InfiniteGame | Erzeugt nach und nach Container und Schiffe. Steuert alle Spielevents. |

# Zusammenarbeitsdiagramme

## Systemoperation: deployContainer

### Beschreibung

Diese Systemoperation beschreibt den Vorgang, wie ein Container mittels des Krans vom Zug auf das Schiff transportiert wird. Ausgelöst wird der Vorgang von den Touch-Eingaben des Benutzers.

#### Hinweise

Der Kran soll, wenn die Methode deployContainer(Container container) aufgerufen wurde und er so das Containerobjekt erhalten hat, sich selbstständig zu den Zielkoordinaten bewegen und anschliessend das Containerobjekt dem Schiff übergeben. Realisiert wird das mit Actions, einem Konzept des libGDX-Frameworks. Eine Action beschreibt eine Aktion, die über eine gewisse Zeitdauer automatisch ausgeführt wird. In diesem Fall wird eine Aktionssequenz zusammengestellt, bestehend aus einer MoveToAction (bewege dich zu Punkt (x,y)), einer von uns definierten Action, welche die addContainer(Container container)-Methode auf dem Schiff aufruft, und anschliessend eine weitere MoveToAction (bewege dich zurück zum Ausgangspunkt). Diese Aktionssequenz wird dann von libGDX eigenständig abgearbeitet. Deshalb ist der letzte Aufruf asynchron.

### Sequenzdiagramm

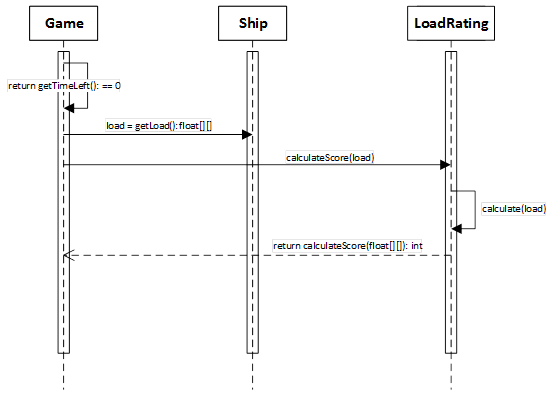


## Systemoperation: calculateScore

### Beschreibung

In dieser Systemoperation wird der Punktestand des soeben gespielten Spiels berechnet. Ausgangslage ist, dass das Spiel soeben beendet wurde. Der Timer für die verbleibende Zeit hat soeben 0 erreicht. Die Game Klasse holt den aktuellen Ladungsverteilungs-Array von der Schiffs-Klasse und gibt diesen für die Berechnung des Punkte Standes an die LoadRating-Klasse weiter. Diese ruft intern die Methode Calculate auf, welche auch während des Spiels benötigt wird. Aus den errechneten breakValue, capsizeValue und beautyValue wird dann der Punktestand berechnet und mit dem Handicap-Faktor multipliziert und zurückgegeben.

### Sequenzdiagramm



## Systemoperation: saveGame

### Beschreibung

Die Operation saveGame wird vom CareerGame nach erfolgreicher Beendigung eines Levels ausgeführt. Sie setzt die neuen Daten im Statistics-Objekt (vermutlich ein Singleton, da es sehr ähnliche Aufgaben wie eine Log-Klasse hat) und ruft anschliessend die Methode save() auf, welche das Statistics-Objekt dazu veranlasst, alle Daten über die Persistenzschicht zu speichern.

#### Hinweis

Die Statistik- und Persistenzfunktionen sind noch nicht vollständig modelliert und dementsprechend auch nicht im Klassendiagramm vorhanden. Ausserdem werden die Spielstandsdaten vermutlich noch von den Statistikdaten entkoppelt. Anstatt :Statistics wird wohl in Bezug auf den Spielstand in Zukunft von :GameState oder ähnlichem die Rede sein.

### Kommunikationsdiagramm

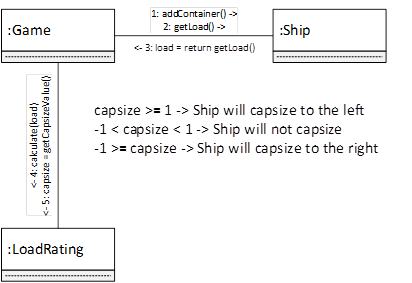


## Systemoperation: capsizeShip

### Beschreibung

In dieser Systemoperation wird nach jedem versetzten Container die Ladung neu berechnet um zu eruieren, ob das Schiff bald kentert. Die errechneten Werte könnten in einer Art Wasserwaage dargestellt werden, um den Spieler beim Beladen visuell zu unterstützen. Nach jedem verladenen Container, wird der aktuelle Ladungsverteilungs-Array des Schiffs abgefragt, und der LoadRating-Klasse zur Berechnung übergeben. Nach der Berechnung, wird der capsizeValue abgefragt, aus welchem ersichtlich ist, ob das Schiff kentert. Werte grösser gleich 1 bedeuten, das Schiff kentert nach links. Werte kleiner gleich -1 bedeuten, das Schiff kentert nach rechts.

### Kommunikationsdiagramm



## Systemoperation: breakShip

### Beschreibung

Die Systemoperation breakShip prüft mittels der LoadRating-Klasse, ob die Ladung des Ship-Objekts ausgeglichen beladen ist. Ist die Ladung an einem Ort schlecht verteilt (d.h. der breakValue ist höher als 1), soll das Schiff brechen (eine Animation wird abgespielt).

### Kommunikationsdiagramm

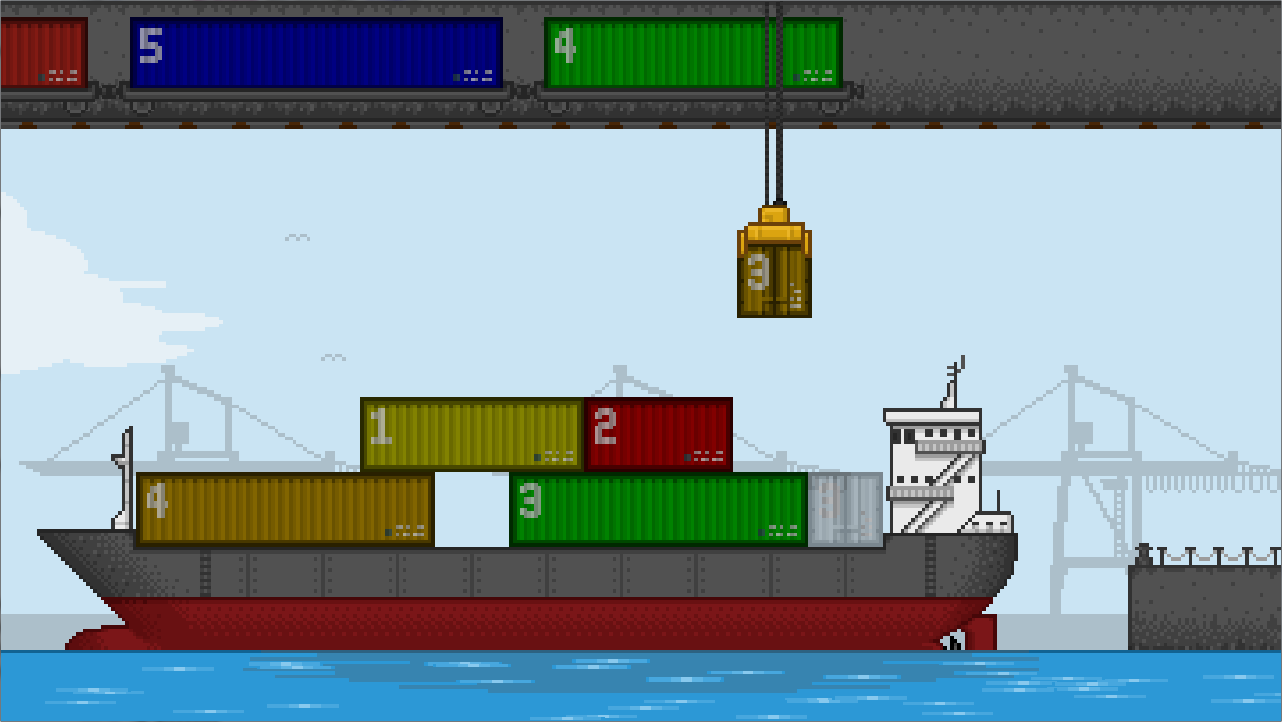


# GUI-Design

## Menüs

Unsere Menüstruktur ist sehr simpel und flach. Ausserdem wollen wir uns in dieser Phase noch auf den Spielablauf konzentrieren, weshalb das Menü optisch noch nicht gestaltet wurde. Generell werden wir im Umfang des SEPS-Projekts das Menü nicht priorisiert behandeln und werden es erst verschönern, wenn sich die zeitliche Gelegenheit dazu bietet.

## Spielbildschirm



Der Spielbildschirm hat sich sehr stark am ursprünglichen Mock-Up orientiert und ist bereits fast in einem auslieferungsfähigen Zustand. Einige Ecken und Enden müssen noch ausgebessert werden, etwa die Platzierung der Gewichtszahl muss nach rechts verschoben werden, da man sie so auf dem Zug zu spät sieht.

# Glossar

Dieses Glossar erklärt wesentliche Begriffe des Projekts “Docker”. Folgende Elemente können einen Begriff beschreiben:

|  |  |
| --- | --- |
| **Begriff:** | Der zu erklärende Terminus |
| **Definition:** | Kurze Definition des Begriffs |
| **Weitere Erklärungen:** | Weitere Informationen zum Begriff (optional) |
| **Format:** | Typ, Länge, Einheit (optional) |
| **Validierungsregeln:** | Validierungsregeln für Parameter (optional) |
| **Aliase:** | Synonyme (optional) |
| **Beziehungen:** | Beziehungen dieses Begriffs zu anderen Elementen (optional) |

Diese Auflistung soll nur als Richtlinie dienen. Die meisten Begriffe sollten in kurzer Prosa erklärt werden.

## Projektdomäne

### Primärbegriffe

|  |  |
| --- | --- |
| Spieler | Der Spieler ist der einzige menschliche Akteur in der Projektdomäne.  Synonyme: Benutzer, User, Anwender |
| Spiel | Das Spiel beinhaltet sowohl die Spielregeln und –Logik, als auch den aktuellen Zustand des Spiels, etwa Timer oder die aktuelle Punktezahl. Verschiedene Spielmodi führen zu verschiedenen Ausprägungen des Spiel-Objekts.  Synonyme: Game |
| Schiff | Das Schiff ist ein zentrales Spielelement in Docker. Ziel des Spiels ist es, Container möglichst effizient auf das Schiff zu beladen. Das Schiff enthält also eine Sammlung bereits platzierter Container. Verschiedene Schiffe unterscheiden sich in Attributen wie Breite, Höhe und Tragfähigkeit.  Synonyme: Frachtschiff, Containerschiff |
| Zug | Der Zug ist dafür zuständig, die zu verladenden Container in den Spielbereich zu „liefern“. Abhängig von seiner Geschwindigkeit wird das Spiel einfacher oder schwieriger.  Synonyme: Güterzug, Containerzug |
| Container | Ein Container muss durch den Spieler vom Zug auf das Schiff verladen werden. Container erscheinen in verschiedenen Ausführungen, die sich in Gewicht, Grösse und Farbe unterscheiden können.  Synonyme: Frachtcontainer |
| Kran | Der Kran hat die Aufgabe, Container vom Zug auf das Schiff zu befördern. Dabei bestimmt der Spieler welcher Container auf welche Position gesetzt werden soll. Der Kran führt diese Anweisung dann selbstständig durch.  Synonyme: Hafenkran |
| Level | Ein Level ist eine vordefinierte Spielkonfiguration, die für den Karrieremodus benötigt wird. Es bestimmt, in welcher Reihenfolge welche Container vom Zug in den Spielbereich gebracht werden. Levels sind persistent und werden vom Spiel geladen. |
| Handicap | Das Handicap ist eine Sammlung von Parametern, die das Spiel für den Spieler schwieriger gestalten. Darunter fallen die Geschwindigkeit des Zuges, die Toleranz des Schiffes bezüglich ungleichmässiger Ladung und „blindes Versetzen“. Das Handicap ist persistent und wird vom Spiel geladen. |

### 

### Sekundärbegriffe

|  |  |
| --- | --- |
| Spielbereich | Der Spielbereich ist der „Viewport“, also der Teil des Spiels, der für den Spieler sichtbar auf dem Bildschirm erscheint. So können Objekte theoretisch im negativen Bereich des Spielkoordinatensystems und damit nicht im Spielbereich befinden.  Synonyme: Spielfeld |
| Spielmodus | Der Spielmodus ist eine Variante des Spiels. Während die Grundregeln und -Aufgaben (Schiff beladen) identisch bleiben, beeinflussen sie das Spielerlebnis wesentlich. Im Moment gibt es drei mögliche Spielmodi: Das Schnelle Spiel, das Unendliche Spiel und den Karrieremodus.  Synonyme: Spielvariante |
| Schnelles Spiel | Das schnelle Spiel ist der simpelste Spielmodus. Er beinhaltet ausschliesslich die Grundregeln.  Synonyme: Quick Game  Beziehungen: Use Case „Schnelles Spiel“ |
| Unendliches Spiel | Das unendliche Spiel baut auf dem schnellen Spiel auf, ist aber zeitlich nicht begrenzt. Bloss die ansteigende Schwierigkeit limitiert die Spieldauer. Im Gegensatz zum schnellen Spiel können mehrere Schiffe in Progression beladen werden.  Synonyme: Endloses Spiel, Endless Game, Infinite Game  Beziehungen: Use Case „Unendliches Spiel“ |
| Karrieremodus | Der Karrieremodus teilt das Spielerlebnis in mehrere, vordefinierte Level auf. Ist ein Level geschafft, wird der nächste freigeschaltet.  Synonyme: Career Game  Beziehungen: „Karriere-Modus mit Level-Freischaltung“ |
| Kippwert | Beschreibt, wie nahe das Schiff am Kentern ist.  Synonyme: Kenterwert, capsizeValue  Beziehungen: Systemoperation capsizeShip. Wird von der Klasse LoadRating berechnet |
| Bruchwert | Beschreibt, wie nahe das Schiff am Brechen ist.  Synonyme: breakValue  Beziehungen: Systemoperation breakShip. Wird von der Klasse LoadRating berechnet. |

## Projektmanagement

|  |  |
| --- | --- |
| Höllenquery | Die von RH entworfene Query, die alle Plandaten und Aufwände aus dem Projektmanagement verrechnet. |