Schlussbericht

Gruppe 1

Emily Wangler, Yacine Mekesser, Christoph Mathis, Remo Höppli

08.12.2014

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Inhalt

Anhang

Bedienungsanleitung

Sourcecode auf USB-Stick

Projektmanagement

Das Projektmanagement wurde ein weiteres Mal auf den Stand des Projektabschluss aktualisiert. Das Team konnte das Projekt erfolgreich zu Ende bringen und hat dabei den grössten Teil der definierten Anwendungsfälle umgesetzt. Einige eher nebensächliche Anwendungsfälle, welche wir während des Projekts als unwichtig eingestuft haben, wurden aus dem aktuellen Projekt gestrichen. Diese können jedoch bei einer späteren Fortsetzung der Arbeiten an Docker, dank des modularen Aufbaus, einfach integriert werden. Bei den durch das Team als wichtig erachteten Anwendungsfällen wurde etwas mehr Zeit investiert um die Arbeit nicht nur fertigzustellen, sondern auch qualitativ hochwertig abzuschliessen. Durch das gute Teamwork und die effiziente Arbeitsweise der Mitglieder konnte innert kurzer Zeit ein solides Smartphone-Game entwickelt werden, welches durch eine intuitive Bedienung und charmante Grafik zu begeistern vermag. Ich möchte mich hiermit als Teamleiter auch ganz herzlich bei allen Teammitgliedern für ihren Einsatz und die gute Zusammenarbeit bedanken.

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	D, E			E, EBB, ECC, ECD		E, F
YM	A		A, C, D			E	D, E			E, EBC, ECA, ECAA, ECB		E, EBAA, EBC, F
CM	A		A, C, D			E	D, E			E, EBC, EBCA		E, EBB, EBC, EBCA, F
EW	A		A, C, D			E	D, E			E, EBCB, ECA, ECC		E, ECB, ECC, F
		M1			**M2**						**M3**	**M4**
		P1			**P2**			**P3**				**P4**

Meilensteine: Projektschiene

30.09.2014	Präsentation Projektskizze	P1
21.10.2014	Präsentation Anforderungen	P2
18.11.2014	Präsentationen Design	P3
09.12.2014	Schlusspräsentationen	P4

Meilensteine: Projekt Docker

30.09.2014	Inception Abschluss	M1
21.10.2014	Elaboration Abschluss	M2
02.12.2014	Construction Abschluss	M3
09.12.2014	Transition Abschluss	M4

Legende

Kürzel	Name
RH	Remo Höppli
YM	Yacine Mekesser
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 Level-Generator	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	Statistik Bildschirm	ECC	EW	4.0	4.0	0.0
C3	Projekt	Career Game	EBCB	EW	6.0	5.0	1.0
C3	Projekt	Settings	ECC	EW	3.0	3.0	0.0
C3	Projekt	Level erstellen	ECD	RH	2.0	1.0	1.0
C3	Projekt	Score-Verteilung verfeinern	EBB	RH	3.0	3.0	0.0
C3	Schlusspräsentation	Projektmanagement	E	RH	4.0	2.0	2.0
C3	Projekt	Besprechungen	E	Alle (*4)	12.0	12.0	0.0
C3	Projekt	Game abbrechen mit zurück	EBC	CM	2.0	2.0	0.0
C3	Projekt	Schluss Bildschirm	EBC	CM	2.0	2.0	0.0
C3	Projekt	Infinite Game	EBCA	CM	8.0	6.0	2.0
C3	Projekt	Anzeigen Bruchgefahr	ECAA	YM	3.0	1.0	2.0
C3	Projekt	Anzeigen Kentergefahr	ECAA	YM	3.0	2.0	1.0
C3	Projekt	Abstract Game	EBC	YM	3.0	3.0	0.0
C3	Schlusspräsentation	Anleitung	E	RH	3.0	2.0	1.0

C3	Schlusspräsentation	Zusammenfassung	E	EW	2.0	1.0	1.0
C3	Schlusspräsentation	Test	E	CM	2.0	0.0	2.0
C3	Schlusspräsentation	Klassendiagramm	E	YM	1.0	0.0	1.0
C3	Projekt	Javadoc	E	Alle (*4)	12.0	12.0	0.0
C3	Projekt	Rendering	ECA	YM	4.0	4.0	0.0
C3	Projekt	Persistence	ECA	EW	2.0	2.0	0.0
C3	Projekt	Score Bildschirm	ECC	RH	2.0	2.0	0.0
C3	Projekt	Menu	ECB	YM	2.0	2.0	0.0
T1	Projekt	Refactoring Abstract Game	EBC	CM	1.0	1.0	0.0
T1	Projekt	Refactoring Load Rating	EBB	CM	1.0	1.0	0.0
T1	Projekt	Infinite Game	EBCA	CM	2.0	2.0	0.0
T1	Projekt	Refactoring Menu	ECB	EW	2.0	3.0	1.0
T1	Projekt	Refactoring Statistik	ECC	EW	1.0	1.0	0.0
T1	Schlusspräsentation	Anleitung	E	RH	1.0	1.0	0.0
T1	Schlusspräsentation	Zusammenfassung	E	EW	2.0	2.0	0.0
T1	Schlusspräsentation	Test	E	CM	2.0	2.0	0.0
T1	Schlusspräsentation	Klassendiagramm	E	YM	1.0	1.0	0.0
T1	Schlusspräsentation	Projektmanagement	E	RH	2.0	2.0	0.0
T1	Projekt	Besprechungen	F	Alle (*4)	12.0	10.0	2.0
T1	Projekt	Advertisements	E	EW	2.0	2.0	0.0
T1	Projekt	Credits	E	EW	2.0	2.0	0.0
T1	Schlusspräsentation	Einführung Präsentation	F	RH	3.0	3.0	0.0
T1	Schlusspräsentation	Erweiterungen Präsentation	F	CM	3.0	2.0	1.0
T1	Projekt	Abstract Game Animation	EBC	YM	2.0	2.0	0.0
T1	Projekt	Ship Logik Ladehöhenindex	EBAA	YM	2.0	2.0	0.0
T1	Projekt	Allgemeine Abschlussarbeiten	F	Alle (*4)	12.0	12.0	0.0

Stundenerfassung

Aufwände

Name	I1	E1	C1	C2	C3	T1	Total
Remo Höppli	12	12	13	14.5	16	11.5	79
Yacine Mekesser	14	12	15	10.5	18	10.5	80
Christoph Mathis	10	10	12	15.5	16	13.5	77
Emily Wangler	10	10	6	11.5	21	15.5	74
Total	46	44	46	52	71	51	310

Prognose

Name	I1	E1	C1	C2	C3	T1	Total
Remo Höppli	12	12	11	17	20	12	84
Yacine Mekesser	14	12	12	14	22	11	85
Christoph Mathis	10	10	12	16	20	15	83
Emily Wangler	10	10	10	15	23	15	83
Total	46	44	45	62	85	53	335

Differenz (verfügbare Stunden)

Name	I1	E1	C1	C2	C3	T1	Total
Remo Höppli	0	0	-2	2.5	4	0.5	5
Yacine Mekesser	0	0	-3	3.5	4	0.5	5
Christoph Mathis	0	0	0	0.5	4	1.5	6
Emily Wangler	0	0	4	3.5	2	-0.5	9
Total	0	0	-1	10	14	2	25

Gesamtprognose	**400**
Bisher benötigt	**310**
Verbleibend	**90**

Risiken

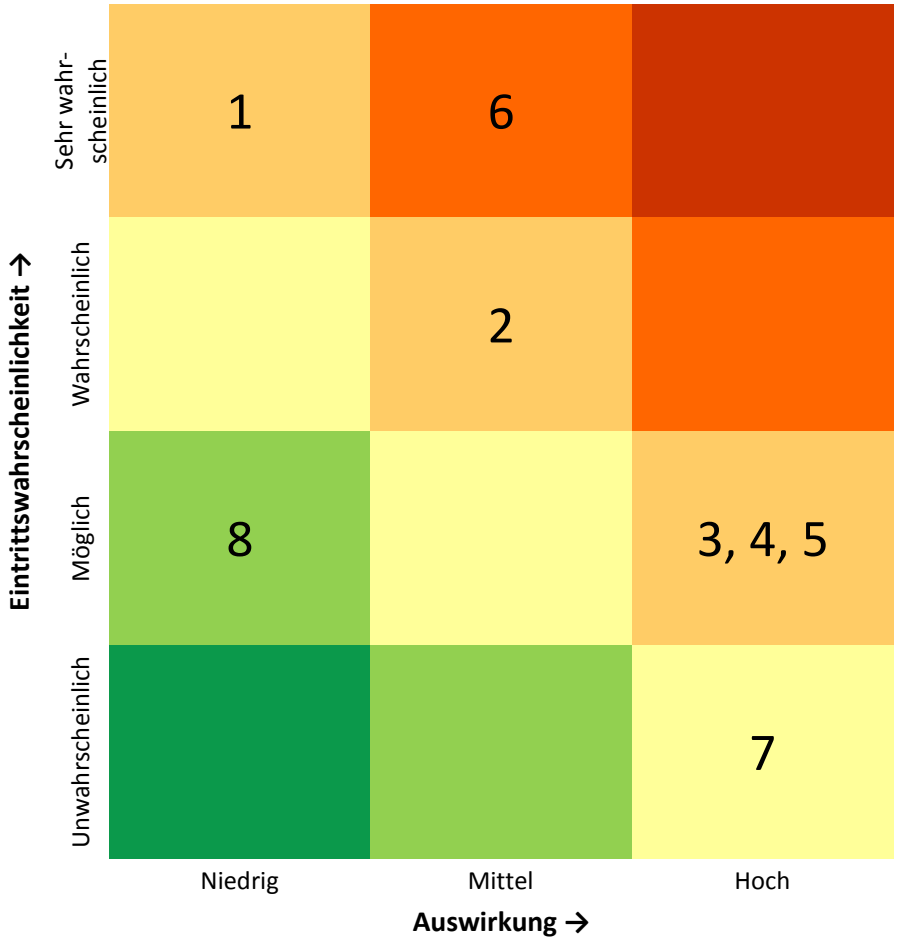
Nr.	Risiko	Beschreibung	EW	AW	Massnahmen
1	ZHAW Netzwerk	ZHAW Server sind aufgrund eines Wartungsfensters oder Ausfalls nicht erreichbar.	Sehr wahr-scheinlich	Gering	Git benutzen.
2	Motivation	Motivation während des Semesters lässt nach.	Wahr-scheinlich	Mittel	Arbeiten gerecht verteilen. Teamgeist pflegen 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

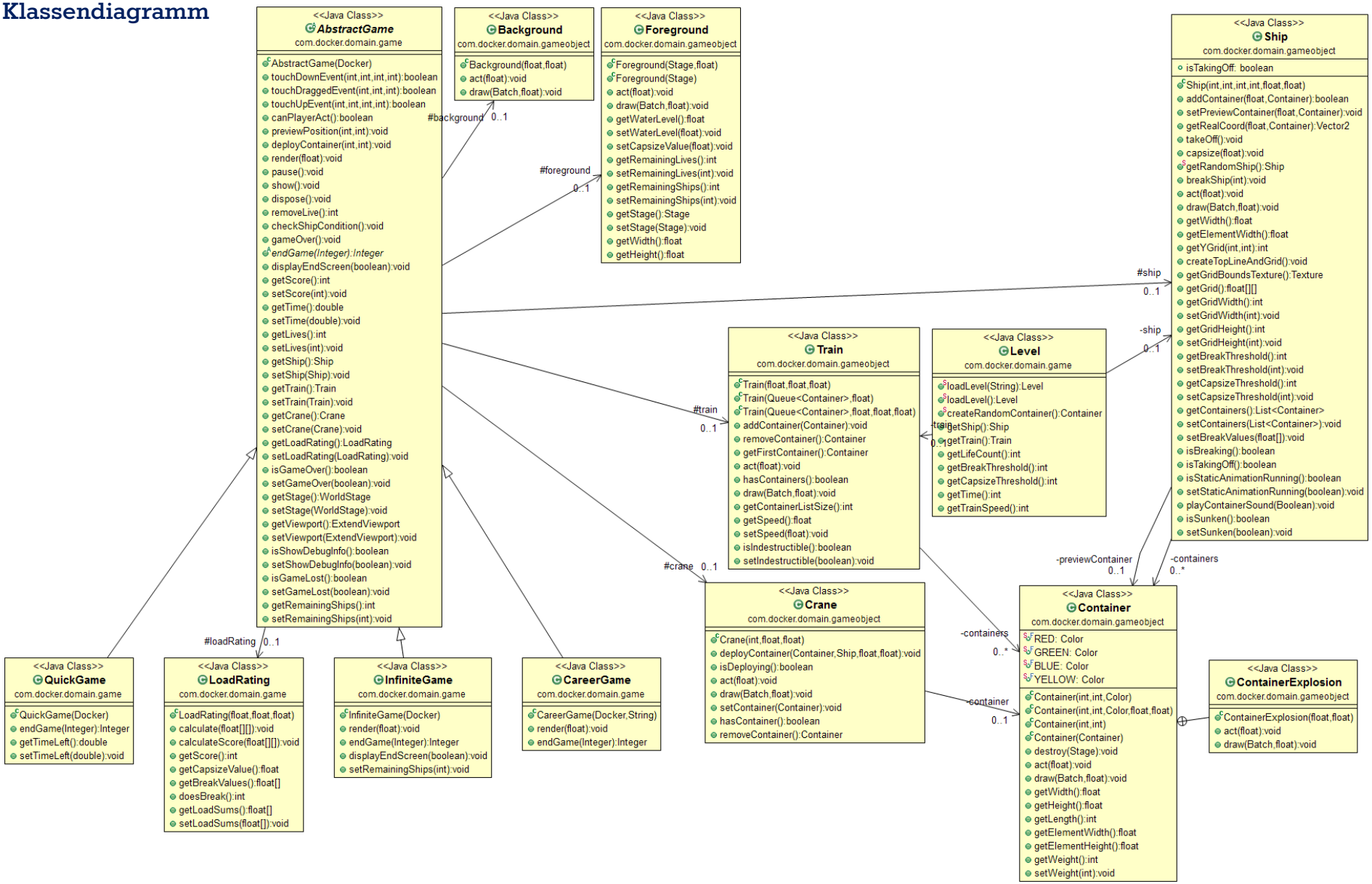
Besonderes

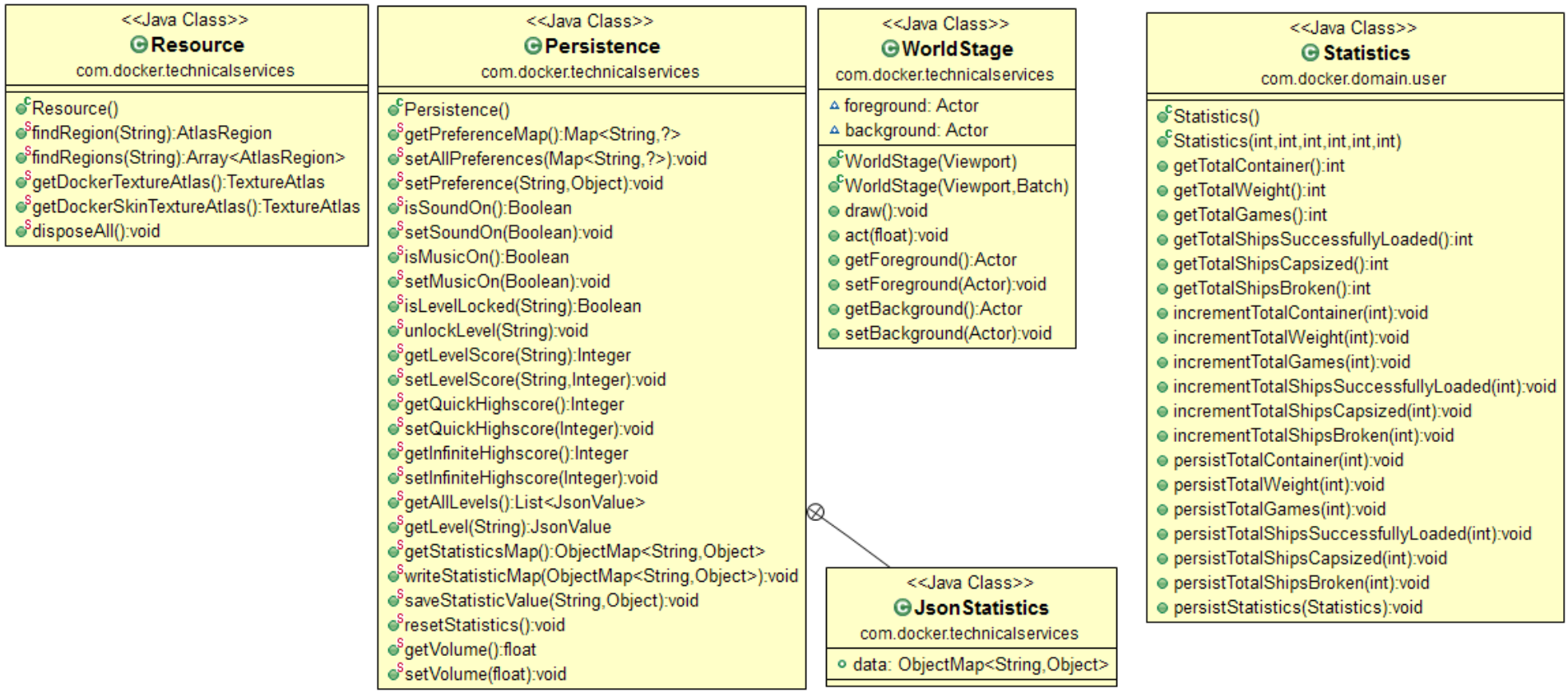
Obwohl das Risiko mit der Nummer 6 während des Projektes zwei Mal in Richtung höherer Gefahr angepasst werden musste, ist das Projekt sehr erfolgreich verlaufen. Nicht zuletzt ist dies den zu Beginn definierten Massnahmen zu verdanken, welche während des Projektes geholfen haben die Auswirkungen durch die definierten Risiken zu minimieren. Natürlich haben auch der Einsatz der einzelnen Teammitglieder und die flexible Planung seinen Teil zum erfolgreichen Verlauf des Projektes beigetragen.

Risikodiagramm



Klassendiagramm




<<Java Class>>
Resource
com.docker.technicalservices
Resource()
findRegion(String):AtlasRegion
findRegions(String):Array<AtlasRegion>
getDockerTextureAtlas():TextureAtlas
getDockerSkinTextureAtlas():TextureAtlas
disposeAll():void
<<Java Class>>
Persistence
com.docker.technicalservices
Persistence()
getPreferenceMap():Map<String,?>
setAllPreferences(Map<String,?>):void
setPreference(String,Object):void
isSoundOn():Boolean
setSoundOn(Boolean):void
isMusicOn():Boolean
setMusicOn(Boolean):void
isLevelLocked(String):Boolean
unlockLevel(String):void
getLevelScore(String):Integer
setLevelScore(String,Integer):void
getQuickHighscore():Integer
setQuickHighscore(Integer):void
getInfiniteHighscore():Integer
setInfiniteHighscore(Integer):void
getAllLevels():List<JsonValue>
getLevel(String):JsonValue
getStatisticsMap():ObjectMap<String,Object>
writeStatisticMap(ObjectMap<String,Object>):void
saveStatisticValue(String,Object):void
resetStatistics():void
getVolume():float
setVolume(float):void
<<Java Class>>
WorldStage
com.docker.technicalservices
foreground: Actor
background: Actor
WorldStage(Viewport)
WorldStage(Viewport,Batch)
draw():void
act(float):void
getForeground():Actor
setForeground(Actor):void
getBackground():Actor
setBackground(Actor):void
<<Java Class>>
JsonStatistics
com.docker.technicalservices
data: ObjectMap<String,Object>
<<Java Class>>
Statistics
com.docker.domain.user
Statistics()
Statistics(int,int,int,int,int,int)
getTotalContainer():int
getTotalWeight():int
getTotalGames():int
getTotalShipsSuccessfullyLoaded():int
getTotalShipsCapsized():int
getTotalShipsBroken():int
incrementTotalContainer(int):void
incrementTotalWeight(int):void
incrementTotalGames(int):void
incrementTotalShipsSuccessfullyLoaded(int):void
incrementTotalShipsCapsized(int):void
incrementTotalShipsBroken(int):void
persistTotalContainer(int):void
persistTotalWeight(int):void
persistTotalGames(int):void
persistTotalShipsSuccessfullyLoaded(int):void
persistTotalShipsCapsized(int):void
persistTotalShipsBroken(int):void
persistStatistics(Statistics):void

Testbericht

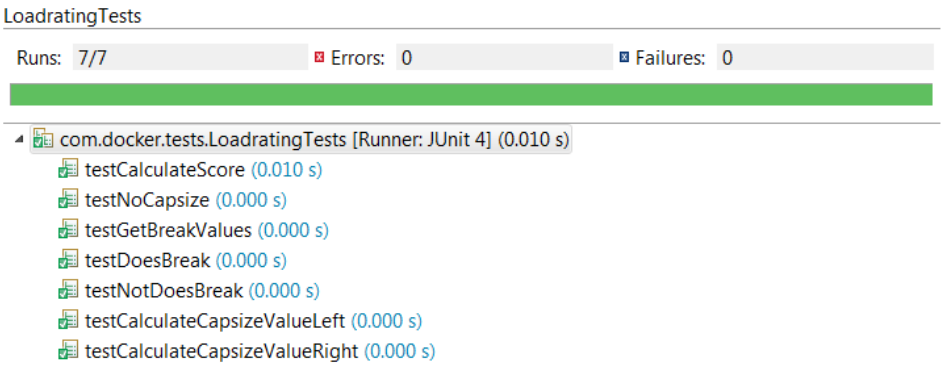
Beim Testen haben wir uns auf die Klasse LoadRating konzentriert, weil sie den Ausgang des Spieles entscheidet und dadurch eine sehr zentrale Klasse ist. Wir haben Junit Blackbox Tests geschrieben in welchen wir die verschiedenen Ausgänge eines Spieles simuliert haben. Der Klasse LoadRating wurde zu Beginn jedes Tests neu initialisiert, damit die vorhergehenden Tests den neusten nicht beeinflussen.

Bei der Eingabe wird immer ein Array mit Ladungswerten übergeben welche die Gewichtsverteilung auf dem Schiff darstellen.

Was getestet wird	Äquivalenzklassen	Bedeutung	Eingabe	Ausgabewert
Der Schlusscore soll richtig berechnet werden.	1. 0-2999 2. 3000	1. Schiff wurde nicht perfekt beladen 2. Schiff wurde perfekt beladen	Es wird ein perfekt geladenes Schiff übergeben	Das Ergebnis war eine Score von 3000.
Es wird ermittelt ob das Schiff kentert.	1. -unendlich bis -1 2. -0.99 bis 0.99 3. 1 bis unendlich	1. Das Schiff kippt nach rechts 2. Schiff kippt nicht 3. Schiffkippt nach links	Hier werden für alle 3 Fälle je einen Test durchgeführt.	Je nach Test haben wir alle verschiedenen Werte erhalten.
Es wird ermittelt ob das Schiff bricht.	1. -1 2. 0 – unendlich	1. Das Schiff bricht nicht 2. Der Wert gibt an, an welcher Position das Schiff bricht.	Hier wurden zwei verschiedene Schiffe eingegeben, eines das eine Bruchstelle hat und eines welches nicht bricht.	Das Schiff ohne Bruchstelle lieferte wie erwartet eine -1. Das Schiff mit der Bruchstelle lieferte eine 2.

Wir haben getestet, ob der Schlussscore richtig berechnet wird, anhand eines perfekt beladenen Schiffes. Wir haben mit verschiedenen Container Anordnungen getestet ob das Schiff sinken wird und falls es sinkt, auf welche Seite es kentert. Zudem haben wir geschaut ob die verschiedenen Bruchwerte eines Schiffes richtig berechnet werden und ob die Bruchposition richtig ermittelt wird.

Die Tests liefen alle Automatisch und erfolgreich ab, wie diesem Bild entnommen werden kann.



Zusammenfassung der erreichten Ziele

Funktionalität

Im Spiel „Docker“ muss man möglichst geschickt Container auf ein Frachtschiff laden und dabei sowohl deren Länge sowie auch Gewicht beachten, sodass der Frachter am Ende nicht untergeht oder gar zerbricht. Dazu wurden drei verschiedene Spielmodi umgesetzt:

- **Quick Game**: Das schnelle Spiel hat eine fixe Anzahl Container, welche auf das Schiff geladen werden müssen.
- **Infinite Game**: Hier werden immer neue Container generiert und die Schiffe können selber weitergeschickt werden wenn sie voll sind, so dass ein neues erscheint.
- **Career Game**: Im Karriere-Modus gilt es die vordefinierten Container erfolgreich auf die Schiffe zu verladen, um das nächste Level frei zu schalten.

Neben dem Hauptspiel bietet das Android-App folgende Funktionen:

- **Settings**: In den Einstellungen können die Soundeffekte und die Musik ein- und ausgeschalten sowie die Lautstärke gesteuert werden.
- **Statistics**: Neben dem Highscore werden hier viele interessante Statistiken angezeigt, zum Beispiel wie viele Container gesamthaft beladen wurden oder die Anzahl verlorener Schiffe.
- **Credits**: Hier werden alle, die am Projekt mitgewirkt hatten oder sonst dazu beigetragen haben, aufgelistet.

Fehler / Einschränkungen

Um das Projekt vollständig abzuschliessen müssen noch einige kleinere Fehler behoben werden. Im Moment sind noch folgende Tickets ausstehend:

- **Errorhandling für Preview Container #29**: Wenn ein Previewcontainer angezeigt wird, dann der Container wechselt und der neue kein Platz auf dem Schiff hat, wird eine Exception geworfen. Dies sollte man Erkennen und korrekt behandeln.
- **Absturz im Infinite Game #39**: Wenn im Infinite Game das Schiff am Untergehen ist, kann in einem kurzen Moment immer noch ein Container gesetzt werden. Dies führt zu einem Absturz des Spiels.

Nicht Umgesetzt

Folgende Punkte wurden im Verlaufe des Projektes trotz Planung nicht umgesetzt.

- Handicap-Einstellungen: Um die Schwierigkeit zu steuern, kann der Spieler hier z.B. die Zugsgeschwindigkeit verstellen. Dieses Menu wurde tiefer priorisiert und wegen Zeitmangel nicht umgesetzt.
- In-App-Käufe: Der Spieler sollte die Möglichkeit haben, sein Spiel mit verschiedenen In-App-Käufen aufzuwerten oder die Werbung auszuschalten. Dieser Use-Case war von Anfang an ein Wunsch und wurde klar ans untere Ender der Prioritätenliste gesetzt.
- Tutorial: Obwohl das Spiel möglichst intuitiv gestaltet wurde, sollte es dennoch eine kleine, interaktive Anleitung für den Benutzer geben. Im Moment ist dies nur in Form der Bedienungsanleitung gegeben.

Rückblick (Erfahrungen)

Die wertvollste Erfahrung, welche wir im Verlauf dieses Kurses gemacht haben, war dass wir eigenständig ein Projekt von Anfang bis Ende durchgeführt hatten. Dies ist im Arbeitsalltag eher selten möglich. Auch konnten wir unsere eigenen Ideen umsetzen und gestalten, was die Arbeit kreativer gestaltete, als dies in einem Unternehmen möglich ist.

Aus technischer Sicht haben wir viele neue Erfahrungen in der Android-Entwicklung gesammelt. Ausserdem lernten wir das Framework libGDX kennen und befassten uns auch etwas mit OpenGL. Auch war die Game-Entwicklung selber interessant, da dort viele Aspekte für uns neu waren und Patterns anders angewendet werden, als dies in üblichen Businessapplikationen der Fall ist. Auch in Bezug auf das Gamedesign selber konnten wir einiges lernen.

Im Team sind wir über anfängliche Spannungen hinweggekommen und konnten gemeinsam und effizient unser Ziel verfolgen und erreichen.

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. Anders als im 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.

Sonstiges

Achievements	Virtuelle Errungenschaften, die für den Spieler freigeschaltet werden, wenn er bestimmte Ziele erreicht (z.B. 100 Schiffe beladen). Achievements können auch mit Belohnungen gepaart werden (z.B. neue Schiffe freigeschaltet werden). Synonyme: Errungenschaften Beziehungen: Use Case Achievements
In-App Käufe	Die Möglichkeit, im Spiel gegen kleine Transaktionen (Microtransactions) spezielle Inhalte freizuschalten. Beispielsweise spezielle Schiffstypen oder Spielhilfen wie etwa die Möglichkeit, den Zug zu verlangsamen. Beziehungen: Use Case In-App Käufe