Analyse Auftrag Docker

Inhalt

Projektmanagement

Neu wurde ein Projektstrukturplan erstellt, in welchem die einzelnen Arbeitspakete pro Person und Iterationen ersichtlich sind. Die Grobplanung aus der Projektskizze wurde aktualisiert, wird aber in künftigen Dokumenten auf Grund von redundanter Information nicht mehr vorhanden sein. Die bisherigen Ziele wurden alle erreicht, weshalb keine weiteren Massnahmen für die nächste Iteration geplant werde müssen. Die weiteren Iterationen wurden im Detail geplant. Die Risiken-Planung musste aktualisiert werden, da effektiv ein Personaldefizit auftreten wird. Yacine Mekesser muss vom 25.11 bis zum 12.12 in den Wiederholungskurs. Seine Abwesenheit ist bereits in die Planung zukünftiger Iterationen eingeflossen und sollte das Projekt nicht gefährden.

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

E Implementation
- EA Domain
  - EAA GameObjects
  - EAB QuickGame
  - EAC InfiniteGame
  - EAD CareerGame
- EB User Interface
  - EBA Rendering
  - EBB Menu
  - EBC User Config & Stats
  - EBD Level
- EC Tech. Services
  - ECA 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	AA, AD, AEA		AEB, CA, DA, DAA, DB			AEB, EAB		AEB, EAB, EAD		AEB, EAD, EBD, F		G
YM	AA, AC, BA		BA, CA, DA, DB, DBA			EBA, ECA		EBA, EAC				
CM	AA, AF		DB, CA, CAA, CAB			EAB, EBC		EAB, EBB, EBC		EBB, EBD, EAD, F		G
EW	AA, AB		CA, CB, DA, DB			EAA, EBA, ECA		EBA, EAC, F		EAC, EAC, F		G
		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	Präsentation Design	P3
09.12.2014	Schlusspräsentation	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

Aufwände

	I1	**E1**	**C1**	**C2**	**C3**	**T1**	**Total**
RH	12	12					24
YM	14	12					26
CM	10	10					20
EW	10	10					20
Total	46	44	0	0	0	0	90
					Prognostiziert		400
					Verbleibend		310

Überarbeitung Grobprojektplanung

Phase	Iteration	Ziele	Nicht erreichte Punkte
Inception	I1	~~Projektskizze erstellt, IDE eingerichtet, erste Ausformulierung der Anwendungsfälle, erster Entwurf der Architektur~~	-
Elaboration	E1	~~Anwendungsfälle definiert, Architektur und Domänenmodell, GUI Designkonzept/Prototyp erstellt, Android-Programmierung Engineering~~	-
Construction	C1	siehe Projektstrukturplan und Softwareentwicklungsplan	
Construction	C2	siehe Projektstrukturplan und Softwareentwicklungsplan	
Construction	C3	siehe Projektstrukturplan und Softwareentwicklungsplan	
Transition	T1	siehe Projektstrukturplan und Softwareentwicklungsplan	

Iterationsdauer: 1 -3 Wochen

Risiken

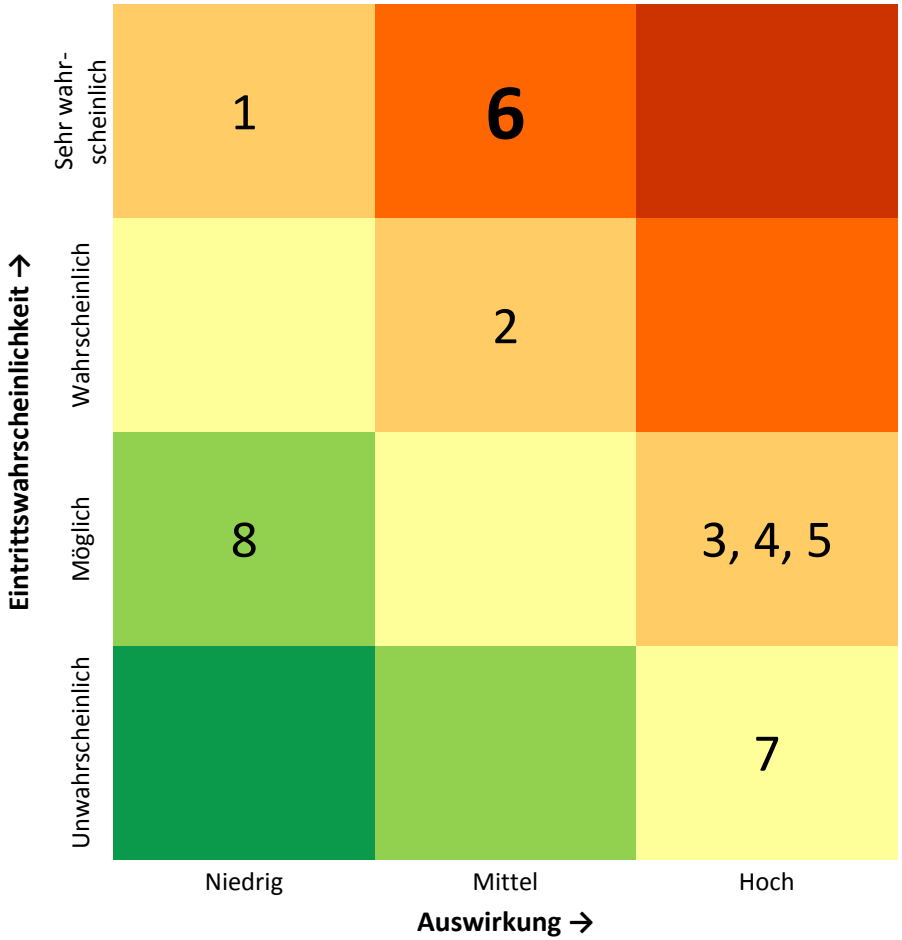
Nr.	Risiko	Beschreibung	EW	AW	Massnahmen
1	ZHAW Netzwerk	ZHAW Server sind aufgrund eines Wartungsfensters oder Ausfalls nicht erreichbar.	Sehr wahr-scheinlich	Gering	Git benutzen.
2	Motivation	Motivation während des Semesters lässt nach.	Wahr-scheinlich	Mittel	Arbeiten gerecht verteilen. Teamgeist pflegen 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!**	**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 Nr. 6 wird im Verlaufe des Projektes eintreten, da Yacine Mekesser vom 24.11 bis zum 12.12 in den WK (Wiederholungskurs) muss.
Als Massnahme wurde die Construction Phase um eine Iteration von 2 Wochen verlängert und die Transition Phase von 3 Wochen auf 1 Woche verkürzt.
Weitere Änderungen in der Personaleinplanung werden während des Projektverlaufs aktualisiert.

Risikodiagramm



Anwendungsfälle

Fully Dressed: Schnelles Spiel

Use Case: Schnelles Spiel

Umfang: „Docker“, Android Anwendung/Spiel
Ebene: Anwenderziel
Primärakteur: Spieler
Stakeholder und Interessen:

- Spieler: Möchte spontan und in einer begrenzten Zeitspanne (z.B. in einer kurzen Pause oder während einer Busfahrt) ein Spiel spielen. Das „Schnelle Spiel“ hilft ihm auch, einen schnellen, ersten Eindruck vom Spiel zu erhalten.

Vorbedingungen: Der Spieler hat die Anwendung geöffnet.
Nachbedingungen: Die Spielstatistiken sind gespeichert und der Highscore wurde mit dem Punktestand des Spiels aktualisiert. Der Spieler ist wieder im Hauptmenü der Anwendung.
Standardablauf:

1. Der Spieler wählt im Hauptmenü „Schnelles Spiel“.
2. Innert 10s (oder weniger) lädt das Spiel.
3. Der Spieler befindet sich nun im regulären Spielbildschirm. Der (interne) Timer steht auf 60s. Es verbleiben 3 Versuche.
4. Im oberen Bildschirmabschnitt bewegt sich ein Güterzug mit Containern von links in das Spielgeschehen hinein. Ab diesem Moment beginnt der (interne) Timer herunter zu zählen.
5. Der Spieler bestimmt, wo er den vordersten Container des Güterzugs auf dem Containerschiff platzieren möchte.
6. Das Spiel zeigt eine Animation des Hafenkrans, welcher den Container auf der vom Spieler bestimmten Position ablegt.
7. Das Spiel berechnet die Gewichtsverteilung auf dem Containerschiff neu.
8. Die Schritte 5 bis 7 werden so lange wiederholt, bis der (interne) Timer 0 erreicht. Der Güterzug bringt dann keine neuen Container mehr.
9. Der Spielablauf ist zu diesem Zeitpunkt zu Ende. Das Spiel berechnet anhand der Gewichtsverteilung und Höhe der Containerstapel den Punktestand.
10. Das Spiel zeigt als Schlussanimation an, wie das Containerschiff den Hafen verlässt. Anschliessend wird der erreichte Punktestand angezeigt.

Erweiterungen:

8a. Der vorderste Container auf dem Güterzug erreicht den rechten Bildschirmrand.
1. Eine Animation wird abgespielt, die dem Spieler signalisiert, dass dieser Container verloren ist.
2. Es wird ein Versuch abgezogen.
2a. Handelt es sich um den letzten Versuch, gilt das Spiel als verloren (Game Over) und der Spieler erhält keine Punkte.

10a. Die Gewichtsbelastung auf dem Containerschiff ist zu ungleichmässig.
1a. Falls die Gewichtsbelastung am Bug bzw. Heck des Containerschiffs unverhältnismässig gross ist, wird eine Animation angezeigt, wie das Containerschiff nach vorne bzw. hinten kippt und sinkt.
1b. Falls die Gewichtsbelastung an einem inneren Abschnitt des Containerschiffs unverhältnismässig gross ist, wird eine Animation angezeigt, wie es an der überbelasteten Stelle zerbricht und anschliessend sinkt.
2. Das Spiel gilt als verloren (Game Over) und der Spieler erhält keine Punkte.

Spezielle Anforderungen:

- Die Ladezeit (siehe Schritt 2) beträgt maximal 10 Sekunden.

- Die Sprache ist Englisch
- Die Bedienung erfolgt über Touch-Eingaben und ist möglichst intuitiv.

Liste der Technik- und Datenvariationen: -

Häufigkeit des Auftretens: Erwartungsgemäss sehr häufig. Die Frequenz ist natürlich vom Verhalten des Spielers abhängig. Das „Schnelle Spiel" dürfte hauptsächlich unter folgenden Bedingungen aufgerufen werden:

- Der Spieler hat nur kurz Zeit für ein Spiel und möchte möglichst schnell Einsteigen
- Der Spieler hat kein Interesse am Karriere-Modus
- Der Spieler hat den Karriere-Modus bereits durchgespielt.

Verschiedenes:

Abzuklären / zu definieren:

- Wird der Timer (die ablaufende Zeit) für den Spieler sichtbar dargestellt?
- Kann das Containerschiff auch bereits während dem Spielablauf sinken oder zerbrechen, anstatt erst nach Ablauf der Spielzeit? Die Auswirkung auf das Gameplay wäre, dass der Spieler zu jedem Zeitpunkt im Spiel gezwungen wäre, effizient zu stapeln, anstatt erst am Ende einen effizienten Zustand zu erreichen. Welche Option ein besseres Spielerlebnis bietet, zeigt sich u.U. erst während der Entwicklung / dem Playtesting.

Anmerkungen:

- Hier wird mit 60 Sekunden Spielzeit für das „Schnelle Spiel" gerechnet. Evtl. zeigt sich während der Entwicklung/dem Playtesting, dass eine andere Zeitspanne sinnvoller wäre.

Fully Dressed: Karriere-Modus mit Level-Freischaltung

Use Case: Karriere-Modus mit Level-Freischaltung

Umfang: „Docker", Android Anwendung/Spiel

Ebene: Anwenderziel

Primärakteur: Spieler

Stakeholder und Interessen:

- Spieler: Will seine Fähigkeiten im Spiel verbessern und möglichst viele Levels freischalten um besser zu sein als seine Bekannten.
- Spielhersteller: Will, dass das Spiel möglichst lange gespielt wird, um den Spielern möglichst viel Werbung präsentieren zu können.

Vorbedingungen: Der Spieler hat das Spiel auf seinem Android-Smartphone installiert. Einige Einstellungen im Handicap-Menü sind bereits getätigt worden oder wurden als Standardwerte gesetzt. Das Spiel wurde gestartet.

Nachbedingungen: Der Spielstand des Spielers wird auf seinem Android-Smartphone abgespeichert, so dass er nicht immer wieder von vorne beginnen muss. Der Spieler ist wieder in der Level-Auswahl des Karriere-Modus.

Standardablauf:

1. Der Spieler kann den Karriere-Modus mit Level-Freischaltung direkt im Menü des Spiels starten. Als erstes erscheint eine Übersicht der verschiedenen Levels und Level-Stufen.
2. Zu Beginn ist nur der erste Level in der ersten Level-Stufe verfügbar. Wird dieser Level erfolgreich abgeschlossen ist der zweite Level der ersten Level-Stufe verfügbar. Dies geht iterativ so weitere bis die erste Level-Stufe komplett erfolgreich abgeschlossen wurde. Eine weitere Level-Stufe wird freigeschaltet, sobald der Spieler einen gewissen Punktestand auf seinem Konto hat. Der Kontostand des Spielers errechnet sich aus der Summe des besten Punkteergebnisses jedes einzelnen, erfolgreich abgeschlossenen Levels.
3. Wählt der Spieler ein Level aus, wird das Spiel gestartet. In jedem Level sind die Schiffsgrösse sowie die Container-Abfolge auf dem Zug vordefiniert. Der Spieler muss die

Container der Reihe nach so auf dem Frachtschiff verteilen, dass die Ladung gewichts- sowie volumenmässig möglichst gleichförmig verteilt ist. Sind alle Container verladen, fährt das Schiff los.

4. Nun wird die Ladungsverteilung berechnet und analysiert. Ist die Ladung genügend gleichmässig verteilt, wird aus der Verteilung ein Punktestand errechnet und mit dem Faktor aus dem Handicap-Menü multipliziert. Die vergebenen Punkte werden dem Spieler auf seinem Konto gutgeschrieben.

Erweiterungen:

4a. Wurden die Container während des Spiels nicht genügend gleichmässig auf dem Frachtschiff verteilt kann das Schiff kentern oder auseinanderbrechen. Passiert dies, ist das Spiel beendet und der Level ist nicht erfolgreich abgeschlossen.

3a. Werden die Container während des Spiels zu langsam verladen können Container verloren gehen. Dies passiert, wenn sie auf dem Zug den rechten Bildschirmrand erreichen. Gehen mehr als zwei Container aus zeitlichen Gründen verloren, ist das Spiel beendet und der Level wurde nicht erfolgreich abgeschlossen.

Spezielle Anforderungen:

- Das Handicap-Menü muss implementiert sein.
- Die Bedienung erfolgt über Touch-Eingaben.

Häufigkeit des Auftretens: Laufend

Verschiedenes:

Anmerkungen:

- Aufbau des GUIs ist noch nicht vollständig definiert.

Casual: Statistiken einsehen

Use Case: Statistiken einsehen

Der Spieler will das bisher von ihm im Spiel Erreichte in Zahlen sehen und mit anderen Spielern vergleichen können. Insbesondere der Highscore ist für ihn als Massstab seiner Leistung interessant, der Rest dient eher als unterhaltsame Trivia.

Der Spieler kann im Hauptmenü den Punkt „Statistik“ auswählen. Im Statistikbildschirm werden alle statistischen Werte angezeigt, die in allen bereits gespielten Spielen erfasst wurden. Darunter fallen:

- Highscore für jeden Spielmodus
- Längste Serie im Endlosmodus (Zeitspanne)
- Anzahl gespielter Spiele für jeden Spielmodus
- Anzahl gewonnener Spiele (d.h. das Spiel wurde erfolgreich abgeschlossen)
- Bisherige Spielzeit
- Anzahl verladener Container
- Anzahl beladener Schiffe
- Anzahl zerstörter Schiffe
- Verhältnis von beladenen zu zerstörten Schiffen

Die Liste der zu erfassenden Daten ist weder final noch abschliessend, sie wird natürlich den anderen Anforderungen und dem Spielablauf angepasst. Die Statistik-Funktionen sind niedrig priorisiert. Sollten Werte schwierig zu erfassen sein, dann sollen sie einfach weggelassen werden. Es soll wenig Zeit in die Statistik investiert werden.

Casual: Handicap-Menü

Use Case: Handicap-Menü

Das Handicap-Menü kann durch den Spieler aufgerufen werden um verschiedenste Einstellungen für den Karriere-Modus sowie für das schnelle Spiel vorzunehmen. Dabei hat der Spieler die Möglichkeit die Zuggeschwindigkeit, die Ausgleichstoleranz sowie das Versetzen zu beeinflussen.

Die Zuggeschwindigkeit kann im Menü auf einer Stufe von 1 bis 10 gewählt werden. Wird der Geschwindigkeitslevel erhöht, fährt der Containerzug im Spiel schneller. Dadurch erhöht sich der Schwierigkeitsgrad insofern, dass die Container schneller verladen werden müssen. Als Ausgleich für den erhöhten Schwierigkeitsgrad wird dafür der Faktor, mit welchem die Punkte zum Schluss jedes Spiels multipliziert werden, erhöht.

Die Ausgleichstoleranz für den Containerverlad kann durch den Spieler im Menü auf einer Stufe von 1 bis 10 gewählt werden. Wird der Toleranzlevel gesenkt, kann das Schiff bei einer ungleichmässigen Beladung schneller kentern oder auseinanderbrechen. Dadurch erhöht sich der Schwierigkeitsgrad insofern, dass die Container gewichtsmässig noch besser verteilt werden müssen. Als Ausgleich für den erhöhten Schwierigkeitsgrad wird dafür der Faktor, mit welchem die Punkte zum Schluss jedes Spiels multipliziert werden, erhöht.

Die Ausgleichsanzeige für den Containerverlad kann durch den Spieler im Menü ein- und ausgeschaltet werden. Ist die Ausgleichsanzeige eingeschaltet, werden dem Spieler während des Spiels kritische Ladungsverteilungen optisch signalisiert. Wird die Ausgleichsanzeige ausgeschaltet, fehlen diese optischen Hilfsmittel und der Spieler ist selbst für die Berechnung der Ladungsverteilung verantwortlich. Als Ausgleich für den erhöhten Schwierigkeitsgrad wird dafür der Faktor, mit welchem die Punkte zum Schluss jedes Spiels multipliziert werden, erhöht.

Die Funktion „blindes Verladen“ kann durch den Spieler im Menü ein- und ausgeschaltet werden. Ist die Funktion deaktiviert, wird dem Spieler während des Spiels die zukünftige Position des Containers bereits vor dem Verladen mittels eines gestrichelten Container-Umrisses angezeigt. Wird die Funktion aktiviert, wird dem Spieler die zukünftige Position des zu verladenden Containers nicht mehr angezeigt. Als Ausgleich für den erhöhten Schwierigkeitsgrad wird dafür der Faktor, mit welchem die Punkte zum Schluss jedes Spiels multipliziert werden, erhöht.

Casual: Tutorial

Use Case: Tutorial

Damit ein neuer Spieler einen einfachen Einstieg ins Spiel bekommt, kann er das Tutorial machen. Das Tutorial erklärt mit einem einfachen Spiel den Spielablauf:

Zu Beginn ist nur der Kran sichtbar auf dem Spielfeld. Der Spieler wird durch Pfeile angewiesen den Kran durch Berühren des Bildschirmes zuerst an den rechten, danach an den linken Bildschirmrand zu bewegen. Hat er dies erfolgreich ausgeführt, erscheint das Schiff auf dem Spielfeld und der Kran bekommt einen Container.

Der Kran fährt (ohne Spielereinfluss) an eine bestimmte Stelle und setzt den Container ab. So sieht der Spieler, was zu tun ist. Mit einem neuen Container am Kran und mit Hilfe von Pfeilen platziert der Spieler den Container am vorgegeben Ort. Setzt er ihn an einem anderen Ort ab, wird er automatisch wieder entfernt und an den Kran gehängt.

Nach erfolgreichem Beladen dieses ersten Containers erscheint sogleich der Güterzug im oberen Bildschirmbereich. Ein kurzer Text erklärt den Sinn des Zuges, welcher bereits mit dem ersten

Container beladen ist. Wieder wird der Spieler angewiesen, den neuen Container auf dem Schiff zu platzieren.

Nun hat er die Möglichkeit das Tutorial weiter zu spielen, bis das Schiff voll ist. Während des ganzen Tutorials wird dem Spieler gezeigt, wo er die Container hinstellen soll, damit das Schiff am Schluss korrekt beladen ist. Oder er kann jederzeit das Tutorial abbrechen und mit dem normalen Spiel beginnen.

Brief: Einstellungen vornehmen

Use Case: Einstellungen vornehmen

Der Spieler wählt im Hauptmenü den Punkt „Einstellungen“. Dort bekommt er die Möglichkeit, mehrere Optionen, die das Spielerlebnis beeinflussen, auszuwählen. Diese wirken sich nicht direkt auf das Gameplay bzw. die Spielmechanik aus, sondern drehen sich eher um die Darstellung und das Interface. Die Einstellungen werden bei einer Änderung sofort gespeichert. Es gibt keinen separaten Speichervorgang, -button oder ähnliches.

Brief: Unendliches Spiel

Use Case: Unendliches Spiel

Wenn der Spieler, das unendliche Spiel als Spielmodus wählt, beginnt das Spiel mit dem Einfahren des Zuges, welcher die Container ans Dock bringt. Ein Schiff, welches vom Spieler beladen werden soll, steht schon bereit.

Nun soll der Spieler die Container auf das Schiff beladen, bis er denkt, dass genug Container auf dem Schiff sind und das Schiff losfahren kann. Dann trifft ein neues Schiff an, welches er wiederum beladen soll. Dabei treffen immer neue Container auf dem Gleis ein, die verladen werden müssen. Ab einer gewissen Punktzahl werden die Container schneller eintreffen und es wird schwieriger, alle Container rechtzeitig so auf das Schiff zu verladen, dass es nicht kentert.

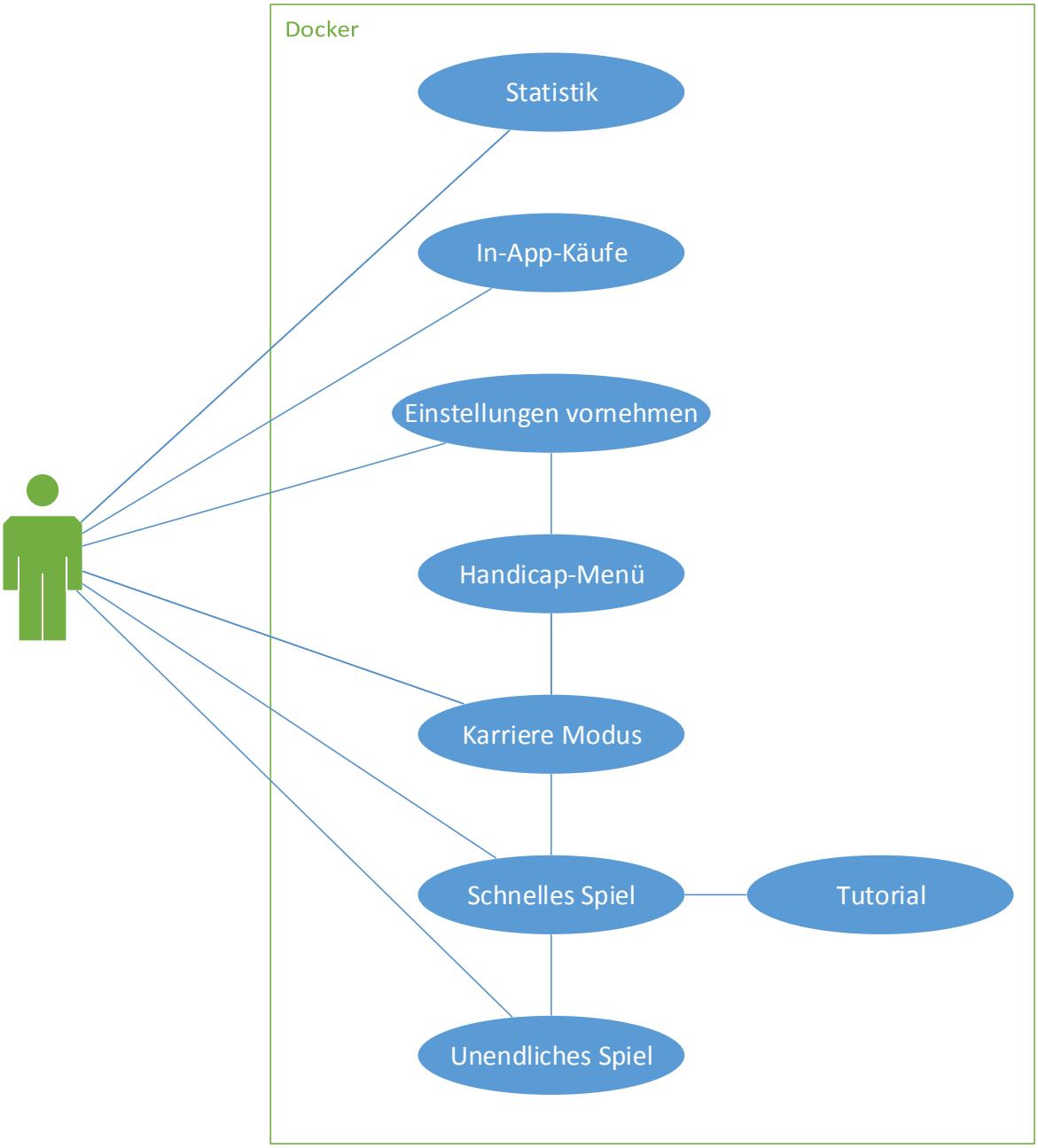
Das Spiel wird nicht durch ein Zeitlimit beschränkt, falls ein Schiff aber falsch beladen wurde und es untergeht wird das Spiel beendet und die erreichten Punkte werden dem Spieler angezeigt. Das Spiel kann auch ein Ende nehmen, falls zu viele Container nicht verladen werden und auf dem Gleis den rechten Bildschirmrand erreichen. Nach dem Ende kann der Spieler einen neuen Spielmodus wählen oder mit dem gleichen Modus nochmal ein Spiel beginnen.

Brief: In-App-Käufe

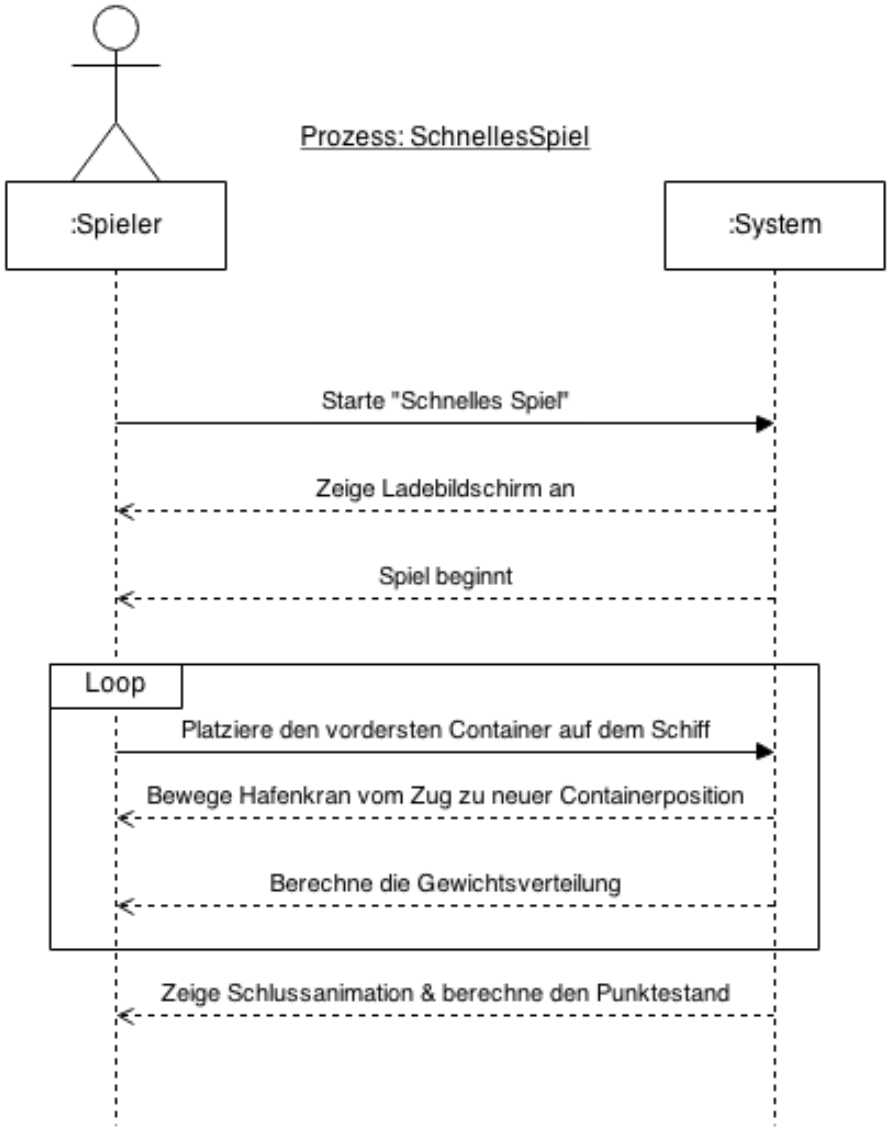
Use Case: In-App-Käufe

Der Spieler kann im Hauptmenü den Punkt „In-App-Käufe“ auswählen. Dort werden ihm verschiedene Spielmodifikationen angeboten, welche das Spiel erleichtern oder das allgemeine Spielerlebnis aufwerten (z.B kann die Werbung aus der App entfernen werden). Eine Modifikation, welche das Spiel erleichtert ist zum Beispiel die Möglichkeit, den Zug für eine kurze Zeit anzuhalten, so dass mehr Zeit gewonnen wird. Nachdem der Spieler diese Modifikationen gekauft hat, werden ihm diese auf seinem Konto gutgeschrieben und sind direkt für ihn verfügbar.

Anwendungsfalldiagramm



System-Sequenzdiagramm



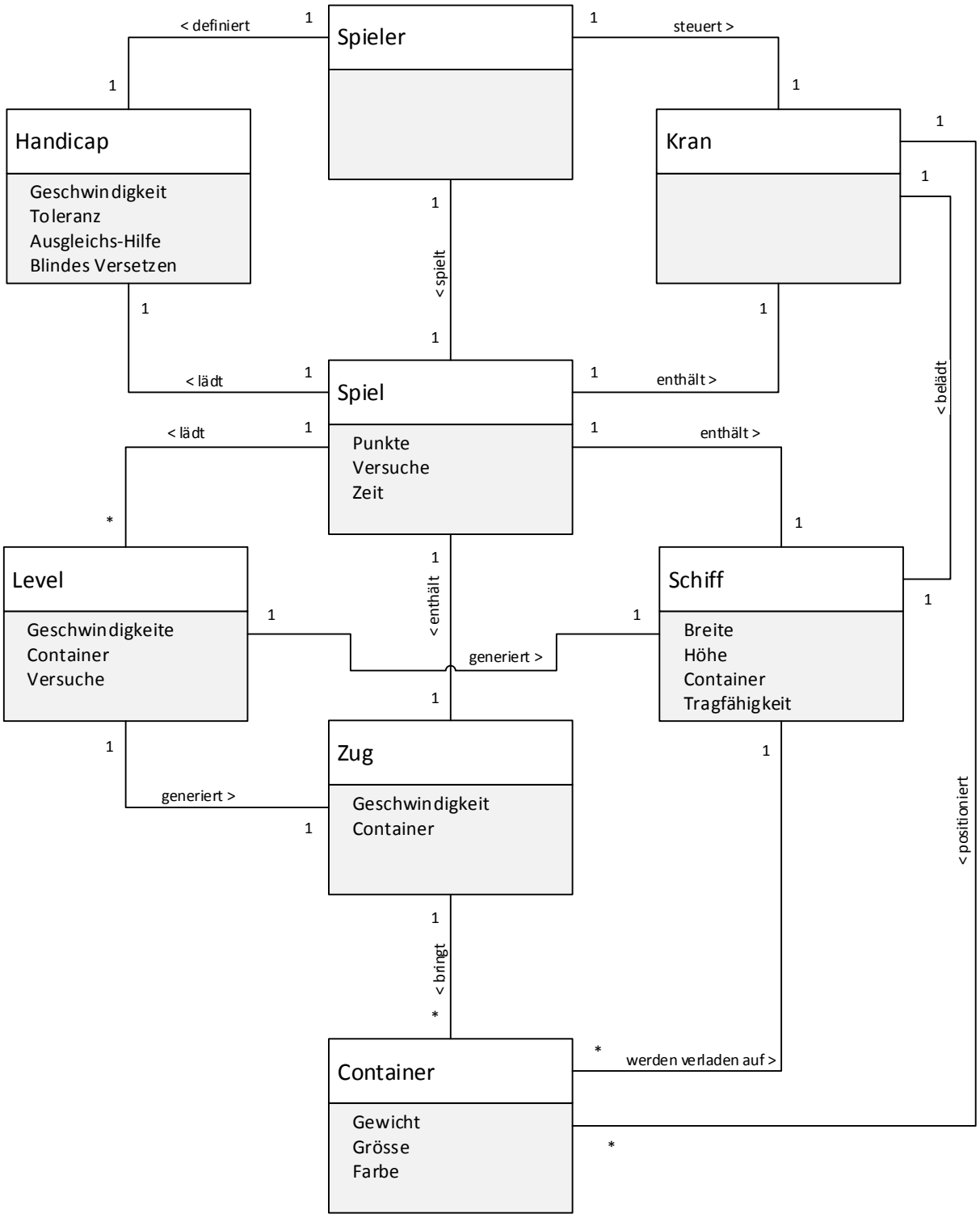
Systemoperationen

Contract CO1: starte „Schnelles Spiel“	
Operation:	starteSchnellesSpiel
Querverweise:	Use Cases: Schnelles Spiel
Vorbedingungen:	Der Spieler hat die Anwendung geöffnet.

Nachbedingungen:	• Ressourcen wurden geladen
	• Das Spiel ist gestartet, d.h.
	  ○ Zug bewegt sich
	  ○ Interner Timer läuft

Contract CO2: platziere den vordersten Container auf dem Schiff	
Operation:	platziereContainer(position)
Querverweise:	Use Cases: Schnelles Spiel
Vorbedingungen:	Das Spiel ist gestartet und es sind zu platzierende Container im Bildschirm.
Nachbedingungen:	• Der Platz auf dem Schiff ist von diesem Container belegt • Der Hafenkran hat den Container an der vorgegebenen Stelle platziert. • Die Gewichtsverteilung wurde neu berechnet.

Domänenmodell



Erste Architektur

Allgemeine Überlegungen

Im Zentrum unserer Architektur-Überlegungen stand zuerst die Frage, ob und wie wir die grafische Repräsentation unserer Spielobjekte von deren Logik und Zustand trennen. Das führte uns zum klassischen Model-View-Controller-Pattern. Nach Absprache mit unserem Dozenten kamen wir aber zum Schluss, dass dies nicht dem angestrebten Domain-Driven Design entsprach und zu einer unnötig künstlichen Trennung der Domänenobjekte führte.

Die Architektur richtet sich nun stark nach dem Domänenmodell und führt Spielobjekte ein, die sowohl einen inneren Zustand haben als auch Logik bereitstellen. Bewusst getrennt werden allerdings die Renderer, mit dem Ziel genug Abstraktion zu erreichen, damit die visuelle Darstellung der Spielobjekte leicht ausgetauscht werden kann. Ausserdem werden die Renderer selber ein gewisses Mass an Logik beinhalten müssen (z.B. Handling von Animationen, Zusammensetzen von Tilesets), die aber nicht direkt mit der Domäne zusammenhängt.

libGDX

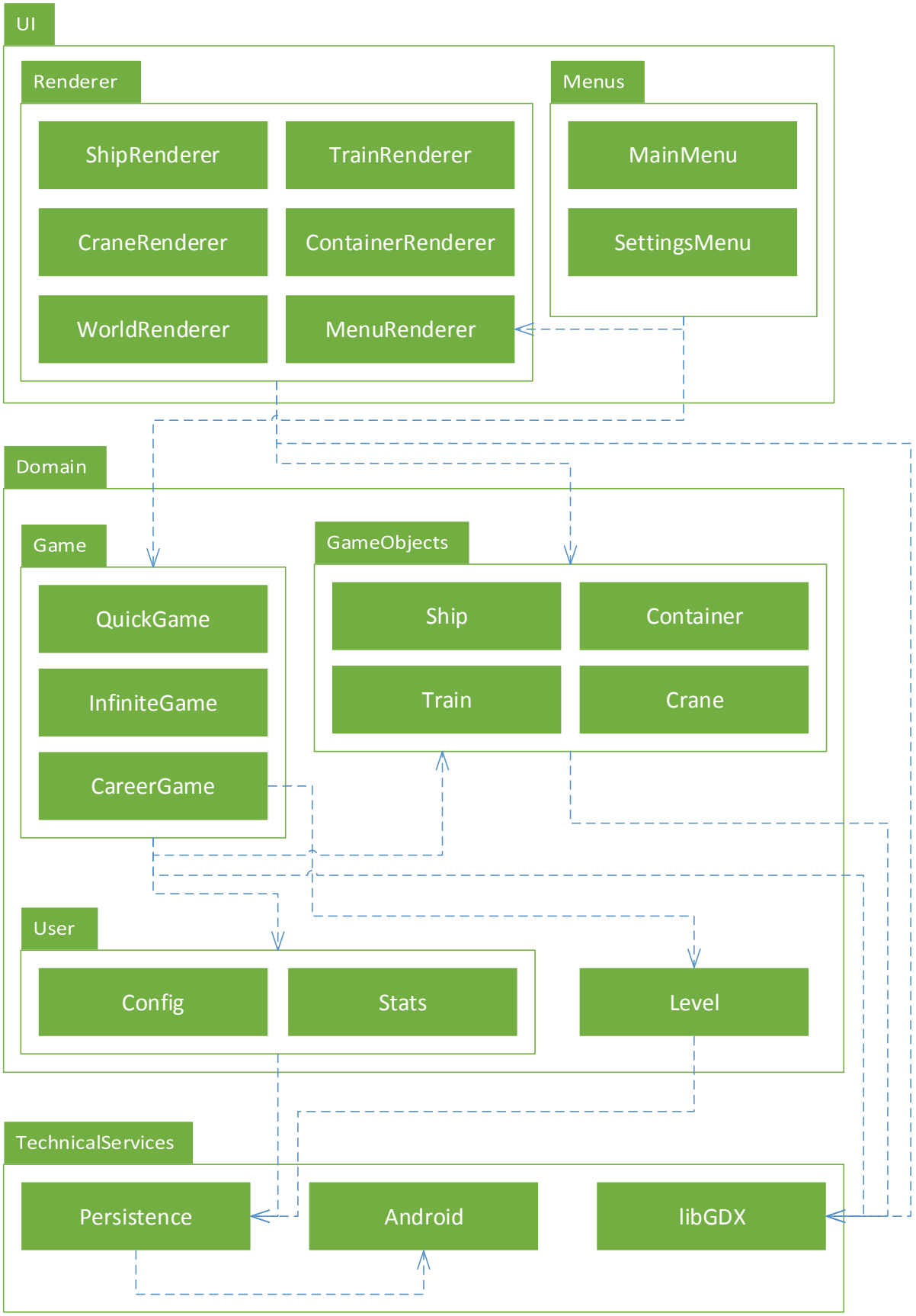
Für die Spieleentwicklung auf Android suchten wir ein Einsteigerfreundliches, aber effizientes Spieleentwicklungsframework. Nach der Analyse von vielen Optionen fiel unsere Wahl auf das libGDX (http://libgdx.badlogicgames.com/) Framework. Die Gründe für diese Entscheidung sind in erster Linie folgende:

- Open Source
  - libGDX steht unter der Apache 2.0 Lizenz.
- Viele, modulare Features
  - Bietet alles, was die Entwicklung eines Spies unseres Scopes erfordern (und mehr).
  - APIs zu Bibliotheken wie z.B. Box2D für 2D-Physikberechnung sind optional.
- In aktiver Entwicklung
  - Version 1.4.1 ist erst kürzlich (10.10.14) erschienen.
- Grosse, aktive Community
  - libGDX hat sein eigenes Forum, einen IRC Channel.
  - Auch z.B. auf StackOverflow und Reddit finden sich viele libGDX-Entwickler, in letzterem ist auch der Hauptentwickler von libGDX aktiv.
- Sehr gut dokumentiert
  - Vollständige Javadoc
  - Diverse Tutorials und Erläuterungen
  - Beispielcode und Demos
- Cross Platform
  - Grafikdarstellung durch OpenGL ES 3.0
  - Ermöglicht neben Android auch die Entwicklung für Windows, Mac, Linux, iOS, BlackBerry und HTML5

Pakete

Im Folgenden werden die Pakete (vgl. Paketdiagramm) mit ihren Aufgaben und Beziehungen beschrieben.

Paketdiagramm



UI

Das UI-Paket enthält Pakete, die für die Benutzereingaben und die grafische Darstellung verantwortlich sind.

UI::Renderer

Das Renderer-Paket enthält Objekte (Klassen, evtl. weitere Pakete), die für die grafische Darstellung von Objekten der Spielwelt zuständig sind. So muss beispielsweise der ShipRenderer die Breite des Ship-Objekts auslesen und ist dann dafür verantwortlich, die korrekten Bilddateien zu laden und daraus das Schiff korrekt zusammenzustellen. Der technische Vorteil dieser Trennung ist, dass die grafische Repräsentation von Objekten getrennt ist und so z.B. erst zur Laufzeit bestimmt werden kann, wie ein Objekt gerendert wird (eine Art Dependency Injection). Fachlich macht diese Trennung ebenfalls Sinn, da die Darstellungslogik nichts mit der Domänenlogik zu tun hat. Die Klassen des UI::Renderer-Pakets nutzen die Grafik-APIs von libGDX und greifen auf Objekte des Pakets Domain::GameObjects zu.

UI::Menus

Das Menus-Paket enthält die Logik für die Menüs. Darunter fällt zum Beispiel das Hauptmenü oder das Einstellungs-Menü. Die Menüs haben die Aufgabe, auf Benutzerinputs (z.B. Touch-Input auf ein Button-Objekt) mit den korrekten Aktion (z.B. Schnelles Spiel starten) zu reagieren. Die Klassen des UI::Menus-Pakets nutzen die Klasse(n) aus dem UI::Renderer Paket für die visuelle Darstellung und starten Objekte aus dem Domain::Game Paket.

Domain

Das UI-Paket enthält die Domänenlogik des Projekts. Sein Aufbau richtet sich stark nach dem Domänenmodell.

Domain::Game

Die Objekte im Game-Paket sind für die übergreifende Steuerung der Spiellogik zuständig. Jeder Spielmodus wird von einer eigenen Klasse (oder von einem eigenen Paket, falls mehrere Klassen pro Spielmodus anfallen) definiert. Gemeinsame Operationen werden durch Vererbung generalisiert. Die Objekte im Domain::Game-Paket laden Einstellungen aus den Config- und Stats Klassen aus dem Domain::User-Pakets, im Falle des CareerGames werden zusätzlich auch Level geladen. Die Objekte des Domain::Game-Pakets enthalten Klassen aus dem Domain::GameObjects-Paket und nutzen das Input-Handling von libGDX.

Domain::GameObjects

Die Objekte im GameObjects-Paket repräsentieren die Spielobjekte wie sie im Domänenmodell definiert wurden. Ein GameObject, wie etwa eine Instanz der Ship-Klasse, enthält sowohl Zustandsinformationen als auch Logik. Sie werden im Spiel von den Renderern dargestellt. Für die Benutzerinteraktion wird das Inputhandling von libGDX genutzt.

Domain::User

Das User-Paket enthält Klassen und Funktionalität für die Benutzerspezifischen Einstellungen (etwa das Handicap und spielübergreifende Einstellungen wie Ton an/aus) und Statistiken (wie der Highscore, der Spielfortschritt im Karrieremodus, etc.). Die Daten werden mithilfe des TechnicalServices::Persistence-Pakets gelesen und anschliessend von den Objekten im Domain::Game-Paket verwendet.

Domain::Level

Im Level-Paket befinden sich eine oder mehrere Klassen für das Level-Handling. Ein Level wird über das TechnicalServices::Persistence-Paket geladen und enthält vordefinierte Parameter für ein Spiel, inklusive einer Liste von Containern mit der Reihenfolge wie sie im Spiel erscheinen werden. Die Levels werden ausschliesslich vom CareerGame genutzt.

TechnicalServices

Das TechnicalServices Paket enthält unterstützende, von der Domänenlogik losgelöste und externe Pakete an, die den technischen „Unterboden“ des Projekts bilden.

TechnicalServices::Persistence

Das Persistence-Paket soll als wiederverwendbare Zwischenschicht zwischen den zu persistierenden Objekten (Domain::User::Config, Domain::User::Stats und Domain::Level) und den Android-Bibliotheken dienen. Grund dafür ist, dass Android mehrere Möglichkeiten zur Persistenz anbietet (z.B. Shared Preferences, Internal/External Storage, etc.) und wir noch nicht sicher sind, welche für uns am geeignetsten ist. Ausserdem könnte sich das möglicherweise in Zukunft mit steigenden Speicheranforderungen auch ändern.

TechnicalServices::Android

Die Android-Bibliotheken stellen die API zum Android-Betriebssystem dar und sind essentiell für eine Android-Applikation. In unserem Fall werden wir insbesondere die Storage-Funktionalitäten nutzen, welche von unserem TechnicalServices::Persistence-Paket abstrahiert wird.

TechnicalServices::libGDX

LibGDX ist ein externes Framework (siehe libGDX unter Allgemeine Überlegungen), welches hauptsächlich für das Inputhandling als auch für die Grafikschnittstelle verwendet wird. Es findet in vielen Bereichen unserer Applikation Anwendung, insbesondere im UI::Renderer-Paket, dem Domain::Game-Paket und dem Domain::GameObjects-Paket

Zusätzliche Spezifikationen

Functionality

Für alle Use Cases gelten folgende funktionalen Anforderungen:

System Error Logging

Fehler werden permanent gespeichert und können, wenn vom User so gewünscht, zur Analyse an den Hersteller gesendet werden. Andernfalls kann der User den Fehler auf der Hersteller-Website melden.

Security

Das Programm benötigt keinen Zugriff auf Daten des Mobilgerätes oder des Users, ebenfalls werden keine Kundendaten (Highscores, Einstellungen, etc.) gesammelt oder weitergegeben, sondern nur lokal auf dem Gerät gespeichert.

Usability

- Die Bedienung und die Benutzeroberfläche sollen einfach und intuitiv aufgebaut sein.
- Das Spiel soll in weniger als 5 Minuten von einem geübten Smartphone-User verstanden und gespielt werden können.
- Das Tutorial unterstützt den User beim Verständnis des Spieles.

Reliability

- Das Spiel muss 24 Stunden pro Tag, sieben Tage die Woche verfügbar sein.
- Ebenfalls muss der Standard-Spielablauf jederzeit ohne Fehler gespielt werden können.

Performance

- Da der Spieler meist nur kurze Zeit zum Spielen hat, soll das Spiel in weniger als zehn Sekunden gestartet und spielbar sein.
- Ebenfalls soll es, wegen dem sonst schon begrenzten Speicherplatz auf dem Mobilgerät, weniger als zehn Megabyte Speicher beanspruchen.

Supportability

Die Coding Standards basieren auf den gängigen Java Code Conventions. Besonderen Wert wird jedoch auf folgende Punkte gelegt:

- Code wird in englischer Sprache verfasst
- Ausführliche Java Documentation für jede Klasse ist Pflicht
- Einheitliche Struktur von Klassen und Paketen

Die neuesten Updates mit allfälligen Fehlerbehebungen und Erweiterungen für das Spiel sind über den Google Playstore erhältlich.

Design Constraints

Für die Umsetzung des Spiels werden folgende Plattformen benötigt:

- Als Grundsprache wird Java 1.7 eingesetzt.
- Für die Mobile-Applikation wird das Android SDK verwendet.

- Als Spielframework wird libGDX genutzt.
- Für die Entwicklung wird mit der Entwicklungsumgebung Eclipse gearbeitet.

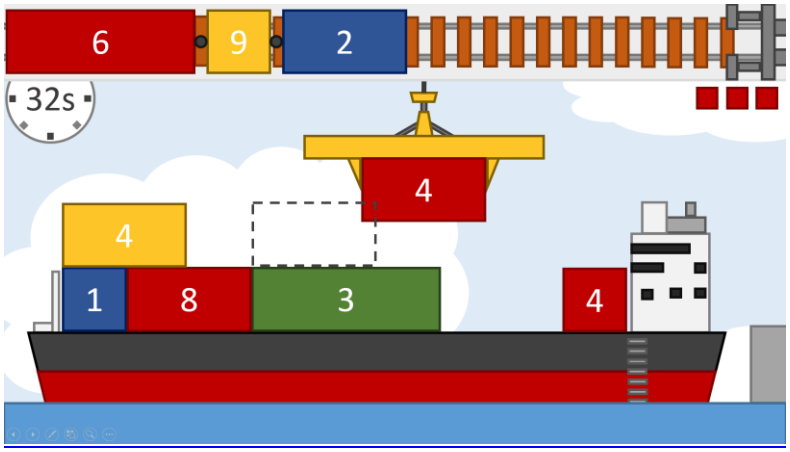
Interfaces

Hardware Anforderungen

- Smartphone mit Touchscreen
- Android ab Version 4.0.0

User Interfaces

- Menu: Damit sich der User schnell zurechtfindet, gibt es ein simples, flaches Menu mit folgenden Punkten: Spiel starten, Einstellungen und Tutorial.
- Spielfeld / Tutorial



- Einstellungen: Hier können folgende Punkte konfiguriert werden:
  - Handicap-Einstellungen
  - Ton ein/aus
  - InApp-Käufe

Lizenzen

Das verwendete Java-Spieleentwicklungsframework libGDX steht unter der Apache 2.0 Lizenz. Diese legt uns unter anderem folgende Pflichten auf, wollen wir libGDX verwenden:

- Es muss eindeutig darauf hingewiesen werden, dass diese Software (libGDX) unter der Apache 2.0 Lizenz steht und diese vom Lizenzgeber stammt.
- Eine Kopie der Lizenz muss dem Paket beiliegen

Im Gegenzug dürfen wir das Softwarepaket kostenlos verwenden, auch für kommerzielle Zwecke. Unsere Software ist nicht gezwungen, die Apache 2.0 Lizenz zu verwenden. Da wir keine Änderungen an libGDX vornehmen werden, betreffen uns die speziellen Bedingungen dazu nicht.

Siehe http://www.apache.org/licenses/LICENSE-2.0.html für die komplette Lizenz.

Die Lizenzbedingungen für das Android SDK finden sich unter https://developer.android.com/sdk/terms.html. Sie schränken unser Projekt nicht ein, da wir weder planen, Änderungen am SDK vorzunehmen, noch mit unserer Applikation gegen geltendes Gesetz zu verstossen.

Urheberrecht

Die Urheberrechte am Spiel stehen gemäss URG Artikel 7 allen Personen, die als Urheber und Urheberinnen an der Schaffung des Werks mitgewirkt haben, gemeinschaftlich zu. In unserem Fall umfasst das alle Mitglieder des Projektteams.

Wir haben keine Rechte an den verwendeten Bibliotheken und Frameworks.

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“