Karlsruher Institut für Technologie

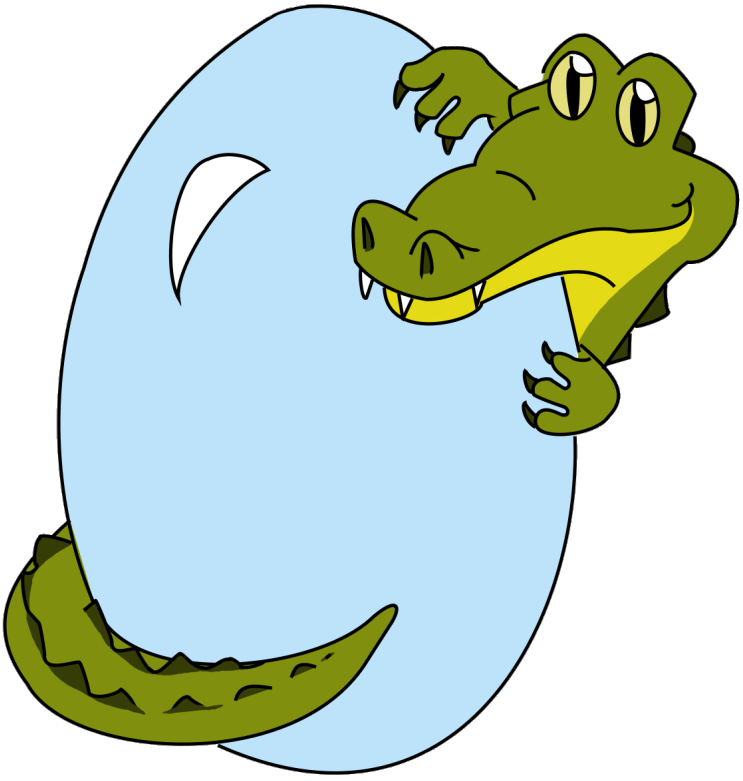
Croggle

Lernanwendung für Grundschüler

Implementierung

Lukas Böhm, Tobias Hornberger, Jonas Mehlhaus,
Iris Mehrbrodt, Vincent Schüßler, Lena Winter

10. Februar 2014



Inhaltsverzeichnis

1 Einleitung

Die Lernanwendung „Croggle“ soll bereits Grundschulkindern einen spielerischen Einstieg in die Konzepte der funktionalen Programmierung bieten, im Speziellen durch den untypisierte Lambda-Kalkül. Auf Grundlage der in der Planung definierten Eigenschaften und deren im Entwurf festgelegten Umsetzung wurde die Applikation nun mithilfe von libgdx für Android implementiert.

Im vorliegenden Implementierungsbericht sind Änderungen zum bisherigen Entwurf, sowie der zeitliche Ablauf und sonstige Ergebnisse der Implementierungsphase dokumentiert. Dazu findet sich zunächst eine detailierte Übersicht über die Umsetzung der in der Planungsphase entwickelten Muss- und Wunschkriterien. Die Änderungen, die zwischen dem Entwurf und der tatsächlichen Implementierung vorgenommen wurden, sind nach Packages aufgeschlüsselt, und können im darauffolgenden Kapitel gefunden werden. Weiterhin wird auf die tatsächliche Umsetzung des am Anfang der Phase festgelegten Zeitplans eingegangen. Zudem wurde das Levelformat gegenüber dem Entwurf angepasst, weshalb die neue Version hier dokumentiert ist. Abschließend folgt eine Übersicht über die Unit Tests.

2 Umsetzung der Muss- und Wunschkriterien

2.1 Musskriterien

Alle Musskriterien konnten vollständig implementiert werden. Zur besseren Übersicht ist hier die Art und Weise der Umsetzung kurz zusammengefasst.

2.1.1 Erstellung und Auswertung von Alligatorenkonstellationen

Es können vorgegebene Alligatorenkonstellationen, je nach Leveltyp, entweder nur umgefärbt, oder durch weitere Objekte vom Spieler ergänzt werden. Die so erstellten Konstellationen können im Simulationsmodus schrittweise ausgewertet werden.

2.1.2 Kontrolle des Lernfortschritts durch Eltern oder Lehrer

Über die erfassten Statistiken können Eltern und Lehrer den Lernfortschritt der Spieler einsehen.

2.1.3 Interaktive Einführung und Erklärung der Regeln

Die Regeln des Spiels sowie die Bedienung der Applikation werden durch Bilder vor den ersten Leveln jeweils erklärt.

2.1.4 Bedienung über ein Tablet mit Toucheingabe

Die Applikation ist vollständig über Toucheingabe bedienbar. Multi-Touch-Gesten wurden implementiert, Croggle ist aber auch ohne Einschränkungen über Single-Touch be-

dienbar. Zudem ist die Applikation auf aktuellen Android Tablets gemäß der Spezifikation im Pflichtenheft lauffähig.

2.1.5 Langzeitmotivation des Spielers wird aufrechterhalten

Über das Achievementsystem und die einfache Erweiterung um neue Level wird eine Langzeitmotivation gewährleistet.

2.2 Umgesetzte Wunschkriterien

Folgende Wunschkriterien konnten implementiert werden:

2.2.1 Speicherung mehrerer Spielstände für verschiedene Benutzer

Die Applikation unterstützt das Anlegen von sechs verschiedenen Benutzern und den Wechsel zwischen diesen. Statistiken, Achievements und Levelfortschritt werden für diese Benutzer jeweils separat gespeichert und erlauben so eine Trennung der Spielerprofile.

2.2.2 Verschiedene Leveltypen zur Verdeutlichung unterschiedlicher Aspekte des Lambda-Kalküls

Es wurden alle drei im Pflichtenheft beschriebenen Leveltypen implementiert. Des Weiteren gibt es Level zum Erlernen von Aussagenlogik im Lambda-Kalküls.

2.2.3 Unterstützung mehrerer Sprachen

Bei der Implementierung wurde auf eine einfache Lokalisierbarkeit geachtet. Bisher ist eine englische und eine deutsche Version vorhanden, Erweiterung um weitere Sprachen sind durch die Erstellung einer weiteren Sprachdatei einfach möglich.

2.2.4 Hilfestellungen für Farbenblinde

Die Anwendung erlaubt die Aktivierung eines „Farbenblindmodus“, wodurch die Farben der Alligatoren und Eier durch zweifarbige, kontrastreiche Muster ersetzt werden. Hierbei fehlt noch eine vollständige Abdeckung aller 30 Farben, jedoch können weitere Muster einfach hinzugefügt werden.

2.3 Nicht umgesetzte Wunschkriterien

Zusätzlich zu den bereits im Entwurf gestrichenen Wunschkriterien wurden in der Implementierung folgende nicht umgesetzt:

2.3.1 Für Smartphones angepasste Version

Bereits die unangepasste, eigentlich für Tablets ausgelegte Version der Applikation stellte sich auch auf Smartphones als überaschend gut nutzbar heraus. Daher wurde keine zusätzliche Arbeit in die Erstellung einer gesonderten Version für kleinere Geräte investiert, auch um den Testaufwand einzuschränken. Zudem ist durch die Nutzung des Model-View-Controller-Musters eine solche Anpassung auch noch im Nachhinein gut möglich und könnte in einer zukünftigen Version umgesetzt werden.

2.3.2 Vermittlung einer Geschichte durch Animationen

Da die Erstellung von weiterem Artwork, insbesondere von Animationen, sehr viel mehr Zeit gekostet hätte, wurde dieses Kriterium nicht implementiert. Das Format zur Beschreibung der Level sieht dies jedoch weiterhin vor, und eine Anzeige der Animationen wäre relativ einfach zu implementieren. Zur Umsetzung fehlen daher hauptsächlich Animationen, die durch das flexible Levelladesystem einfach nachgeliefert werden können.

3 Änderungen gegenüber dem Entwurf

3.1 Root package

3.1.1 Neuerungen in der AlligatorApp

Als Hauptcontroller ist die AlligatorApp nun auch dafür zuständig, das Verhalten des „Back“-Buttons von Androidgeräten zu steuern und verwalten. Dazu enthält sie einen Stack, auf dem bei Bildschirmwechsel die vorige Ansicht abgelegt wird, um sie bei Bedarf erneut aufzurufen. Dazu bietet die Klasse die neuen Methoden:

returnToPreviousScreen Der zuletzt gezeigte Screen wird erneut aufgerufen. Ist dies nicht möglich (z.B. weil der Stack leer ist), wird ein Dialog geöffnet, der es dem Nutzer ermöglicht, die App zu beenden.

showPreviousScreen Der zuletzt gezeigte Screen wird erneut aufgerufen, mit dem Unterschied, dass das „Zurückgehen“ zum vorigen Screen ebenfalls auf dem Stack gespeichert wird; man gelangt also bei erneutem Drücken des „Back“-Buttons zum Ursprungsbildschirm.

clearScreenStack Leert den Stack. Dies ist sinnvoll, wenn durch das „Zurückgehen“ Daten inkonsistent würden oder das Verhalten schlichtweg unerwartet wäre, z.B. nach Gewinnen eines Levels.

Außerdem wurden mehrere Methoden hinzugefügt, die für das Aufrufen von Screens verwendet werden sollten, da in ihnen genaueres Verhalten beim Screenwechsel für die verschiedenen Screens definiert wird, u.a. auch für die Routine beim Drücken des „Back“-Buttons:

- showMainMenuScreen(boolean puOnStack)
- showLevelPackagesScreen()
- showLevelOverviewScreen(LevelController levelController)

- showAcievementScreen()
- showStatisticScreen()
- showSettingsScreen(boolean putOnStack)
- showSelectProfileScreen()
- showProfileSetNameScreen(boolean putOnStack)
- showProfileSetAvatarScreen(String name, boolean putOnStack)
- showPlacementModeScreen(GameController gameController)
- showSimulationModeScreen(GameController gameController)
- showCreditsScreen()
- showLevelTerminatedScreen(GameController gameController)

Es wurde des Weiteren die Methode **created()** hinzugefügt, die aufgerufen wird, sobald die eigentliche create() Methode beendet wurde und alle Daten vom AssetManager fertig geladen sind.

3.2 Package: data

3.2.1 Lokalisierung

Das Paket „data“ wurde lediglich um eine Abstraktionsschicht des Lokalisierungsmechanismus' erweitert. Die dazugehörigen Klassen heißen:

LocalizationBackend Interface, dass grundlegende Methoden zur Lokalisierung bereitstellt.

AndroidLocalizationBackend Implementierung des LocalizationBackend für die Android Plattform

Die Funktionsweise des LocalizationHelpers wurde dahingehend geändert, dass er seine Funktionalität statisch anbietet. Dies ermöglicht die äußerst dezente Anwendung bei der Lokalisierung der Software, da statische Importe verwendet und unnötig lange Bezeichner

vermieden werden können. Die oben bereits genannten Backends werden einmalig zum Programmstart initialisiert.

3.2.2 AssetManager

Der AssetManager stellt eine Erweiterung des von libgdx gelieferten AssetManagers dar. Er kümmert sich zusätzlich um die Verwaltung von zur Laufzeit generierten Texturen, wie sie etwa für die Hintergründe farbiger Spielfeldelemente (BoardObjectActors) benötigt werden.

3.3 Package: data.persistence

3.3.1 LevelProgress

Hinzugefügte Methoden

equals Methode Eine einfache equals Methode wurde hinzugefügt um Testfälle zu vereinfachen.

3.3.2 Setting

Hinzugefügte Methoden

Setting Konstruktor Ein fehlender Konstruktor mit allen für *Setting* relevanten Daten wurde hinzugefügt.

equals Methode Eine einfache equals Methode wurde hinzugefügt um Testfälle zu vereinfachen.

3.3.3 SettingChangeListener

Dieses Interface wurde neu erstellt. Alle Screens die bei einem *Setting*-Wechsel informiert werden müssen dieses Interface implementieren und werden über die *onSettingChange* Methode benachrichtigt.

3.3.4 SettingController

Hinzugefügte Methoden

addSettingChangeListener Über diese Methode können sich Screens, die *SettingChangeListener* implementieren beim *SettingController* anmelden und so über *Setting*-Wechsel informiert werden.

3.3.5 Statistic

Hinzugefügte Methoden

Setting Konstruktor Ein fehlender Konstruktor mit allen für *Statistic* relevanten Daten wurde hinzugefügt.

equals Methode Eine einfache equals Methode wurde hinzugefügt um Testfälle zu vereinfachen.

3.4 Package: data.persistence.manager

3.4.1 AchievementManager

Hinzugefügte Methoden

clearTable Diese Methode löscht alle in der Tabelle gespeicherten Einträge.

getRowCount Diese Methode gibt die Anzahl der in der Tabelle gespeicherten Elemente zurück.

updateUnlockedAchievement Diese Methode modifiziert ein schon in der Datenbank gespeichertes Achievement.

Modifizierte Methoden

deleteUnlockedAchievements Diese Methode wurde entfernt, da das Löschen nun über das Foreign Key ON DELETE statement geregelt wird.

3.4.2 LevelProgressManager

Hinzugefügte Methoden

clearTable Diese Methode löscht alle in der Tabelle gespeicherten Einträge.

getRowCount Diese Methode gibt die Anzahl der in der Tabelle gespeicherten Elemente zurück.

Modifizierte Methoden

deleteUnlockedAchievements Diese Methode wurde entfernt, da das Löschen nun über das Foreign Key ON DELETE Klause statement wird.

3.4.3 PersistenceManager

Hinzugefügte Methoden

clearTables Löscht alle Einträge aus allen Tabellen.

isNameUsed Gibt zurück ob ein gegebener Namen schon von einem in der Datenbank gespeicherten *Profile* benutzt wird.

Modifizierte Methoden

loadProfile Umbenannt in getProfile.

3.4.4 ProfileManager

Hinzugefügte Methoden

clearTable Diese Methode löscht alle in der Tabelle gespeicherten Einträge.

getRowCount Diese Methode gibt die Anzahl der in der Tabelle gespeicherten Elemente zurück.

isNameUsed Gibt zurück ob ein gegebener Namen schon von einem in der Tabelle gespeicherten *Profile* benutzt wird.

3.4.5 SettingManager

Hinzugefügte Methoden

clearTable Diese Methode löscht alle in der Tabelle gespeicherten Einträge.

getRowCount Diese Methode gibt die Anzahl der in der Tabelle gespeicherten Elemente zurück.

Modifizierte Methoden

deleteUnlockedAchievements Diese Methode wurde entfernt, da das Löschen nun über das Foreign Key ON DELETE statement geregelt wird.

3.4.6 StatisticManager

Hinzugefügte Methoden

clearTable Diese Methode löscht alle in der Tabelle gespeicherten Einträge.

getRowCount Diese Methode gibt die Anzahl der in der Tabelle gespeicherten Elemente zurück.

Modifizierte Methoden

deleteUnlockedAchievements Diese Methode wurde entfernt, da das Löschen nun über das Foreign Key ON DELETE statement geregelt wird.

3.4.7 TableManager

Hinzugefügte Methoden

clearTable Diese abstrakte Methode soll alle in der Tabelle gespeicherten Einträge löschen.

getRowCount Diese abstrakte Methode soll die Anzahl der in der Tabelle gespeicherten Elemente zurückgeben.

3.5 Package: game

3.5.1 Color

Hinzugefügte Methoden

uncolored Methode, die die Farbe zurückgibt, die ungefärbten Objekten zugewiesen wird.

getRepresentations Methode, die alle Farbrepräsentationen liefert.

getRepresentation Methode zum Umwandeln einer Farbe im Spiel in eine tatsächlich darstellbare Farbe.

equals Farben sind bei gleicher *id* äquivalent, weshalb *equals* überschrieben wurde.

hashCode Siehe *equals*.

compareTo Siehe *equals*.

3.5.2 ColorController

Hinzugefügte Methoden

getUncolored Liefert die Farbe, die ungefärbten Objekten zugewiesen wird.

Modifizierte Methoden

getRepresentation Umbenannt von *getRepresantation*.

3.5.3 EditLevelGameController

Neu hinzugefügte Klasse, die einen spezielleren *GameController* repräsentiert, der für *EditLevel* genutzt wird.

Hinzugefügte Methoden

onAfterLoadProgress Callbackmethode aus *GameController*. Wird hier genutzt um das letzte *Board* des Benutzers zu laden.

onBeforeSaveProgress Callbackmethode aus *GameController*. Wird hier genutzt um das aktuelle *Board* des Benutzers zu speichern.

createColorController Callbackmethode aus *GameController*. Wird hier genutzt um dem *ColorController* nutzbare sowie gesperret Farben mitzuteilen.

onFinishedSimulation Callbackmethode aus *GameController*. Wird hier genutzt um die Lösung des Benutzers zu validieren.

3.5.4 GameController

Hinzugefügte Methoden

setupColorController Wird zur Re-Initialisierung des *ColorController* aufgerufen.

createColorController Methode, um die Erstellung des *ColorController* durch Unterklassen anpassen zu können.

getColorController Methode, um den *ColorController* zu erhalten.

enterPlacement Methode, die für einen Wechsel in den Platziermodus sorgt. Sichtbarkeit von private auf public geändert.

enterSimulation Methode, die für einen Wechsel in den Simulationsmodus sorgt. Sichtbarkeit von private auf public geändert.

onFinishedSimulation Callbackmethode zum Überschreiben Unterklassen, die nach Abschluss der Simulation aufgerufen wird.

registerSimulationBoardEventListener Siehe *registerBoardEventListener* unter „Entfernte Methoden“.

unregisterSimulationBoardEventListener Siehe *unregisterBoardEventListener* unter „Entfernte Methoden“.

registerPlacementBoardEventListener Siehe *registerBoardEventListener* unter „Entfernte Methoden“.

unregisterPlacementBoardEventListener Siehe *unregisterBoardEventListener* unter „Entfernte Methoden“.

getPlacementBoardEventListener Gibt den *BoardEventMessenger* für den Platziermodus zurück.

onObjectPlaced Neues Event, siehe *BoardEventListener*.

onObjectRemoved Neues Event, siehe *BoardEventListener*.

onObjectMoved Neues Event, siehe *BoardEventListener*.

onAge Neues Event, siehe *BoardEventListener*.

evaluateStep Führt einen Schritt der Simulation aus und prüft, ob diese abgeschlossen ist.

isInSimulationMode Gibt zurück, ob sich der Spieler im Simulationsmodus befindet.

canUndo Gibt zurück, ob ein Schritt in der Simulation rückgängig gemacht werden kann.

undo Macht einen Schritt in der Simulation rückgängig.

reset Setzt das durch den Benutzer bearbeitete *Board* auf den ursprünglichen Zustand zurück.

getShownBoard Gibt das aktuell angezeigte *Board* zurück.

getElapsedTime Gibt die bisher im Level verbrachte Zeit zurück.

getTimeStamp Gibt den Zeitpunkt des Levelstarts zurück.

setElapsedTime Setzt die bisher im Level verbrachte Zeit.

updateTime Aktualisiert die verstrichene Zeit seit dem Levelstart.

setTimeStamp Setzt den Zeitpunkt des Levelstarts.

getLevel Gibt das aktuelle Level zurück.

createPlacementScreen Methode zum Überschreiben durch Unterklassen, um eine andere Darstellung des Platziermodus zu erreichen.

isSolved Gibt zurück, ob das aktuell geladene Level gelöst ist.

getProgress Gibt den *LevelProgress* des aktuellen Levels und Benutzers zurück.

setUserBoard Setzt das durch den Benutzer bearbeitete *Board*.

onBeforeSaveProgress Methode zum Überschreiben durch Unterklassen, die vor dem Speichern des *LevelProgress* aufgerufen wird.

onAfterLoadProgress Methode zum Überschreiben durch Unterklassen, die nach dem Laden des *LevelProgress* aufgerufen wird.

getUserBoard Gibt das durch den Benutzer bearbeitete *Board* zurück.

onUsedHint Muss aufgerufen werden, um die Benutzung eines Hinweises zu registrieren.

getSimulator Gibt den aktuell genutzten *Simulator* zurück.

Modifizierte Methoden

Konstruktor Referenz auf *AlligatorApp* als zusätzliches Argument eingefügt.

onObjectRecolored Geändertes Interface, siehe *ObjectRecoloredListener*.

onEat Geändertes Interface, siehe *EatEventListener*.

onAgedAlligatorVanishes Geändertes Interface, siehe *AgedAlligatorVanishesListener*.

onHatched Umbenannt von *onReplaceEgg*, siehe *Listener*.

Entfernte Methoden

onCompletedLevel Sichtbarkeit von public auf private geändert.

registerBoardEventListener Aufteilung in *registerPlacementBoardEventListener* und *registerSimulationBoardEventListener*.

unregisterBoardEventListener Aufteilung in *unregisterPlacementBoardEventListener* und *unregisterSimulationBoardEventListener*.

3.5.5 MultipleChoiceGameController

Neu hinzugefügte Klasse, die einen spezielleren *GameController* repräsentiert, der für *MultipleChoiceLevel* genutzt wird.

Hinzugefügte Methoden

setSelection Methode, um die Auswahl des Benutzers zu setzen.

createPlacementScreen Überschriebene Methode von *GameController*, um für *MultipleChoiceLevel* eine andere Darstellung zu erzeugen.

3.5.6 Simulator

Hinzugefügte Methoden

canUndo Methode, die zurückgibt, ob aktuell ein Schritt rückgängig gemacht werden kann.

getSteps Methode, die die aktuelle Anzahl an ausgewerteten Schritten zurückgibt.

getCurrentBoard Methode, die das aktuell verwendete *Board* zurückgibt.

Modifizierte Methoden

evaluate Return-Wert von *Board* auf *boolean* geändert. Kann außerdem *ColorOverflowException* und *AlligatorOverflowException* auslösen.

3.6 Package: game.achievement

3.6.1 Achievement

Hinzugefügte Methoden

Getter und Setter Getter für *Index*, *Stage* und *NumberOfStages*, sowie Setter für *Index*, *Description*, *EmblemPathAchieved*, *EmblemPathNotAchieved*, *Id* und *Stages* wurden hinzugefügt.

initialize Abstrakte Methode zur Initialisierung eines *Achievements* wurde hinzugefügt.

Modifizierte Methoden

getEmblemPath Methode wurde aufgespalten in *getEmblemPathAchieved* und *getEmblemPathNotAchieved*.

requirementsMet Methode bekommt jetzt zwei Argumente: *Statistic* statistic und *Statistic* statisticDelta.

3.6.2 AchievementController

Implementiert nun nicht mehr *StatisticsDeltaProcessor*.

Hinzugefügte Methoden

updateAchievements Korrigiert die Indices der *AvailableAchievements*, gibt diese zurück und fügt Änderungen den neusten Änderungen hinzu.

convertInputFromDataBase Konvertiert die Eingabe in eine Liste von *Achievements* mit richtigen Indices und gibt diese zurück.

processStatisticChange Ersetze *processDelta* mit *processStatisticChange*.

Modifizierte Methoden

initializeAvailableAchievements Sichtbarkeit von public zu private geändert.

requirementsMet Entsprechend dem Interface geändert.

3.6.3 AchievementFactory

Neu hinzugefügt. Erstellen von sowohl den einzelnen *Achievements* als auch der *AvailableAchievements* ausgelagert mit dem Factory Entwurfsmuster.

Hinzugefügte Methoden

createAchievement Bekommt eine Id und gibt ein neues, initialisiertes *Achievement* des entsprechenden Typs zurück.

createListOfAchievementTypes Gibt eine Liste mit einem neu initialisiertem *Achievement* von jedem Typ zurück.

3.6.4 AlligatorsEatenAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.5 AlligatorsEatenPerLevelAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.6 AlligatorsPlacedAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.7 AlligatorsPlacedPerLevelAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.8 HintPerLevelAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.9 LevelAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.10 PerLevelAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.6.11 TimeAchievement

Modifizierte Methoden

requirementsMet Entsprechend dem Interface geändert.

3.7 Package: game.board

3.7.1 AgedAlligator

Modifizierte Methoden

Konstruktor Hinzufügen der Flags *movable* und *removable*.

Entfernte Methoden

getParent Bereits in der übergeordneten Klasse *Alligator* implementiert.

setParent Bereits in der übergeordneten Klasse *Alligator* implementiert.

isMovable Bereits in der übergeordneten Klasse *Alligator* implementiert.

isRemovable Bereits in der übergeordneten Klasse *Alligator* implementiert.

3.7.2 Alligator

Hinzugefügte Methoden

isMovable Aus den erbenden Klassen *AgedAlligator* und *ColoredAlligator* verschoben.

isRemovable Aus den erbenden Klassen *AgedAlligator* und *ColoredAlligator* verschoben.

Modifizierte Methoden

Konstruktor Hinzufügen der Flags *movable* und *removable*.

3.7.3 BoardObject

Hinzugefügte Methoden

match Methode, um verschiedene *BoardObject*-Objekte auf Äquivalenz zu testen.

matchWithRecoloring Methode, um verschiedene *BoardObject*-Objekte auf Äquivalenz zu testen, unter Berücksichtigung von möglichen α-Konversionen.

3.7.4 ColoredAlligator

Hinzugefügte Methoden

match Zur Implementierung der Änderungen am Interface *BoardObject*.

matchWithRecoloring Zur Implementierung der Änderungen am Interface *BoardObject*.

Modifizierte Methoden

Konstruktor Hinzufügen der Flags *movable* und *removable*.

3.7.5 ColoredBoardObject

Erweitert nun auch das Interface *InternalBoardObject*.

3.7.6 Egg

Hinzugefügte Methoden

match Zur Implementierung der Änderungen am Interface *BoardObject*.

matchWithRecoloring Zur Implementierung der Änderungen am Interface *BoardObject*.

Modifizierte Methoden

Konstruktor Hinzufügen der Flags *movable* und *removable*

3.7.7 InternalBoardObject

Hinzugefügte Methoden

copy Zur Konkretisierung des Rückgabewertes der Methode aus BoardObject (Kovarianz).

3.7.8 NoSuchChildException

Neu hinzugefügte Exception, die dann ausgelöst wird, wenn versucht wird auf ein nicht vorhandes Kindelement eines *Parent*-Objekts zuzugreifen.

3.7.9 Parent

Implementiert nun das Interface *BoardObject* sowie *Iterable<InternalBoardObject>*.

Hinzugefügte Methoden

insertChild Methode, um ein Kindobjekt an einer festen Stelle einzufügen.

getChildPosition Methode, um die Position eines Kindobjektes zu bestimmen.

getChildAtPosition Methode, um ein Kindobjekt an einer festen Position zu erhalten.

getFirstChild Methode, um das erste Kindobjekt zu erhalten.

getChildCount Methode, um die Anzahl der Kindobjekte zu bestimmen.

clearChildren Methode, um alle Kindobjekte zu entfernen.

acceptOnChildren Methode, die einen *BoardObjectVisitor* entgegennimmt und ihn allen Kindelementen übergibt.

match Zur Implementierung der Änderungen am Interface *BoardObject*.

matchWithRecoloring Zur Implementierung der Änderungen am Interface *BoardObject*.

Modifizierte Methoden

Konstruktor Erweiterung um einen Copy-Konstruktor.

replaceChild Umbenannt von *replaceChildWith*.

iterator Umbenannt von *getIterator*, außerdem Ergänzung um eine Version mit Startindex.

getChildAfter Umbenannt von *getNextChild*.

3.8 Package: game.board.operations (ehemals game.visitor)

Alle Klassen aus dem ehemaligen Paket *game.visitor* wurden entweder in dieses, oder in das untergeordnete Paket *game.board.operations.validation* verschoben. Dabei wurde das Postfix „Visitor“ vom Klassennamen gestrichen. Bei neu hinzugefügten *BoardObject-Visitor* werden im Folgenden die Methoden des Interface nicht extra aufgeführt.

3.8.1 CountBoardObjects

Modifizierte Methoden

count Neue Version hinzugefügt, die eine Konfiguration der gezählten *BoardObject*-Objekte erlaubt.

3.8.2 CreateDepthMap

Neu für den Renderer hinzugefügter *BoardObjectVisitor* um die Höhe der Hierarchie über Objekten zu berechnen.

Hinzugefügte Methoden

create Gibt eine Abbildung von allen Objekten in der übergebenen Hierarchie und ihren berechneten Höhen zurück.

3.8.3 CreateHeightMap

Neu für den Renderer hinzugefügter *BoardObjectVisitor* um die Höhe der Hierarchie unter Objekten zu berechnen.

Hinzugefügte Methoden

create Gibt eine Abbildung von allen Objekten in der übergebenen Hierarchie und ihren berechneten Höhen zurück.

3.8.4 CreateWidthMap

Neu für den Renderer hinzugefügter *BoardObjectVisitor* um die Breite von Objekten, inklusive der von ihren Kindelementen, zu berechnen.

Hinzugefügte Methoden

create Gibt eine Abbildung von allen Objekten in der übergebenen Hierarchie und ihren berechneten Breiten zurück.

3.8.5 ExchangeColor

Neu hinzugefügter *BoardObjectVisitor*, um eine Farbe in einer Familie durch eine andere zu ersetzen.

Hinzugefügte Methoden

recolor Färbt alle Alligatoren und Eier der übergebenen Farbe in der übergebenen Familie durch die zweite übergebene Farbe um.

3.8.6 FlattenTree

Hinzugefügte Methoden

toList Gleiches Verhalten wie *toArray*, nur wird eine Liste anstatt eines Arrays zurückgegeben.

Modifizierte Methoden

toArray Umbenannt von *flatten*. Hinzufügen einer Variante, die nur *InternalBoardObject* arbeitet.

3.8.7 GetParentHierarchy

Neu hinzugefügter *BoardObjectVisitor*, um eine Liste der Elternelemente eines Objektes in der Hierarchie zu ermitteln.

Hinzugefügte Methoden

get Gibt die Liste der Elternelemente des übergebenen Objekts zurück.

3.8.8 RemoveAgedAlligators

Modifizierte Methoden

remove Hinzufügen einer Variante, bei der kein *BoardEventMessenger* übergeben werden muss.

3.8.9 RemoveUselessAgedAlligators

Neu hinzugefügter *BoardObjectVisitor*, um auch solche *AgedAlligator* zu entfernen, die durch die Assoziativität unnötig sind.

Hinzugefügte Methoden

remove Entfernt alle *AgedAlligator* aus der übergebenen Konstellation, die durch die Assoziativität unnötig sind.

3.8.10 ReplaceEggs

Modifizierte Methoden

replace Hinzufügen verschiedene Varianten, bei denen kein *ColorController* oder kein *BoardEventMessenger* übergeben werden muss.

3.9 Package: game.board.operations.validation (ehemals game.visitor)

Alle Klassen die *BoardError* definieren oder behandeln wurden während der Implementierung neu hinzugefügt.

3.9.1 AbstractBoardError

Oberklasse aller *BoardError*. Implementiert das *BoardError*-Interface.

Hinzugefügte Methoden

Konstruktor Es wurde ein Konstruktor hinzugefügt, der ein *BoardObject* als Grund für den auftretenden Fehler übergeben bekommt.

getCause Gibt den Grund, also das *BoardObject* zurück, das den Error verursacht hat.

3.9.2 AbstractBoardValidator

Abstrakte Oberklasse von *ValidateConstellation*.

Hinzugefügte Methoden

Konstruktor

3.9.3 AgedAlligatorChildlessError

BoardError, der auftritt, wenn ein *AgedAlligator* keine Kinder hat.

Hinzugefügte Methoden

Konstruktor Es wurde ein Konstruktor hinzugefügt, der einen *AgedAlligator* als Grund für den auftretenden Fehler übergeben bekommt.

haveDispatched Behandelt den Error auf entsprechende Weise mit Hilfe eines *BoardErrorDispatcher*.

3.9.4 BoardError

Interface, dass alle *BoardError* implementieren. *BoardError* werden benutzt um bei einem invaliden *Board* eine entsprechende, die Ursache des Fehlers enthaltene Fehlermeldung zu erhalten.

Hinzugefügte Methoden

getCause Gibt den Grund, also das *BoardObject* zurück, das den Error verursacht hat.

haveDispatched Behandelt den Error auf entsprechende Weise mit Hilfe eines *BoardErrorDispatcher*.

3.9.5 BoardErrorDispatcher

Interface in dem Methoden definiert werden um die unterschiedlichen Arten von *BoardErrors* zu behandeln.

Hinzugefügte Methoden

dispatch Überladene Methode zur Behandlung unterschiedlicher Arten von *BoardError*.

3.9.6 BoardErrorType

Enum in dem die verschiedenen *BoardError*-Typen definiert werden.

Hinzugefügte Methoden

all Gibt ein Array mit allen in diesem Enum definierten *BoardError*-Typen zurück.

3.9.7 ColoredAlliagtorChildlessError

BoardError, der auftritt, wenn ein *ColoredAlligator* keine Kinder hat.

Hinzugefügte Methoden

Konstruktor Es wurde ein Konstruktor hinzugefügt, der einen *ColoredAlligator* als Grund für den auftretenden Fehler übergeben bekommt.

haveDispatched Behandelt den Error auf entsprechende Weise mit Hilfe eines *BoardErrorDispatcher*.

3.9.8 EmptyBoardError

BoardError, der auftritt, wenn ein *Board* leer ist.

Hinzugefügte Methoden

Konstruktor Es wurde ein Konstruktor hinzugefügt, der ein *Board* als Grund für den auftretenden Fehler übergeben bekommt.

haveDispatched Behandelt den Error auf entsprechende Weise mit Hilfe eines *BoardErrorDispatcher*.

3.9.9 FindBoardErrors

Sucht nach *BoardError* in einem *Board*.

Hinzugefügte Methoden

find Gibt eine Liste an gefundenen *BoardErrors* zurück, die in der übergebenen Alligatorkonstellation auftreten. Je nachdem ob zusätzlich ein *BoardError*-Typ übergeben wurde, werden nur nach *BoardErrors* des übergebenen Typs gesucht.

visit

3.9.10 ObjectUncoloredError

BoardError, der auftritt, wenn ein *ColoredAlligator* oder ein *Egg* nicht eingefärbt ist.

Hinzugefügte Methoden

Konstruktor Es wurde ein Konstruktor hinzugefügt, der ein *InternalBoardObject* als Grund für den auftretenden Fehler übergeben bekommt.

haveDispatched Behandelt den Error auf entsprechende Weise mit Hilfe eines *BoardErrorDispatcher*.

3.9.11 ValidateConstellation

Früher in *board.visitor* als *ValidationVisitor*.

Hinzugefügte Methoden

isValid Diese Methode wurde überladen, damit auch die Suche nach spezifischen *BoardError* möglich ist.

3.10 Package: game.event

Die folgenden Listener wurden zusätzlich zum Entwurf in das „event“ Paket aufgenommen:

AlligatorAgesListener Wenn ein farbiger Alligator frisst, muss er (statt wie in der Entwurfsphase angenommen zu sterben) vorerst altern. Um diesen Vorgang zu kommunizieren, z.B. um eine Animation dazu ablaufen zu lassen, exisitiert das hierzugehörige Event.

BoardEditedListener In der Entwurfsphase wurde vernachlässigt, wie der BoardActor mit Änderungen durch den Nutzer umzugehen hat. Da er auch sonst mit Events arbeitet, war es naheliegend, Änderungen zuerst intern durchzuführen und anschließend das Rendering zu benachrichtigen. Der Listener bietet dazu Methoden zum Empfangen solcher Benachrichtungen.

ReplaceEventListener → EggHatchListener Mit der Einführung der BoardEdited Events wurden allgemein gehaltene Eventnamen für diese reserviert. Durch die Umbenennung entspricht der Name nun auch besser dem genauen Anwendungsszenario.

3.11 Package: game.level

3.11.1 ColorEditLevel

Diese Klasse erbt nun von *EditLevel*.

Modifizierte Methoden

Konstruktor Dem Konstruktor wurden alle Argumente hinzugefügt, die ein Level beschreiben.

3.11.2 EditLevel

Da die Klassen *ColorEditLevel* und *TermEditLevel* viele gemeinsame Methoden und in vielen Fällen eine gleiche Behandlung benötigen, hat es sich angeboten eine gemeinsame Oberklasse einzufügen.

Hinzugefügte Methoden

Konstruktor Dem Konstruktor wurden alle Argumente hinzugefügt, die ein Level beschreiben.

getUserColor Gibt ein Array von Farben zurück, die der Benutzer für dieses Level verwenden kann.

getBlockedColor Gibt ein Array von Farben zurück, die der Benutzer in diesem Level nicnt benutzen darf

3.11.3 InvalidJsonException

Für die einfache Behandlung von Fehlern, die in den Json Dateien der Level auftreten können, hat es sich angeboten, eine eigene Exception für diese Fehlerursache einzufügen.

3.11.4 Level

Hinzugefügte Methoden

getAbortSimulationAfter Gibt zurück nach wievielen Schritten die Simulation beendet werden soll.

getUnlocked Gibt zurück ob das Level schon freigeschaltet wurde.

setUnlocked Mit dieser Methode kann man ein Level auf freigeschaltet bzw. nicht freigeschaltet setzten.

isLevelSolved Gibt zurück ob das übergebene *Board* und Schrittanzahl das Level löst oder nicht.

isSolveable Gibt zurück ob die festgelegte maximale Simulationsschrittzahl bereits überschritten wurde.

isSolved Gibt zurück ob das Level durch das übergebene Board und Schrittanzahl gelöst wird.

setSolvedTrue Setzt ein Level auf gelöst.

createGameController Methode, die einen GameController für dieses Level erstellt und zurückgibt.

getLevelId Methode, die eine Konkatenation von Levelpackage Index und Level Index zurückgibt.

getShowObjectBar Methode, die zurückgibt ob im Plaziermodus für dieses Level die *ObjectBar* angezeigt werden soll oder nicht.

Modifizierte Methoden

Konstruktor Dem Konstruktor wurden alle Argumente hinzugefügt, die ein Level beschreiben.

3.11.5 LevelController

Modifizierte Methoden

Konstruktor Der Konstruktor wurde um einige Argumente ergänzt.

Entfernte Methoden

Konstruktor Überflüssige Konstruktoren wurden entfernt.

3.11.6 LevelPackage

Hinzugefügte Methoden

getAnimation Gibt den Pfad zu der zum Package gehörende Animation zurück.

getDesign Gibt dem Pfad zum Design des Level Pakets zurück.

Modifizierte Methoden

Konstruktor Der Konstruktor wurde um einige Argumente ergänzt.

Entfernte Methoden

Konstruktor Überflüssige Konstruktoren wurden entfernt.

3.11.7 LevelPackagesController

Hinzugefügte Methoden

getLevelController Methode, die zu einem gegebenen *LevelPackage* Index einen *LevelController* zurückgibt.

getLevelPackages Methode, die eine Liste aller vorhandenen Levelpakete zurückgibt.

3.11.8 MultipleChoiceLevel

Modifizierte Methoden

Konstruktor Der Konstruktor wurde um einige Argumente ergänzt.

3.11.9 TermEditLevel

Diese Klasse erbt nun von *EditLevel*.

Modifizierte Methoden

Konstruktor Der Konstruktor wurde um einige Argumente ergänzt.

3.12 Package: game.profile

3.12.1 Profile

Hinzugefügte Methoden

equals Methode Eine simple equals Methode wurde erstellt um Testfälle zu vereinfachen.

3.12.2 ProfileController

Hinzugefügte Methoden

addProfileChangeListener Über diese Methode können sich Screens, die *ProfileChangeListener* implementieren beim *ProfileController* anmelden und so über *Profile*-Wechsel informiert werden.

getCurrentProfile getter für das gerade aktive *Profile*.

getCurrentProfileName getter für den Namen des gerade aktiven *Profile*.

deleteAllProfiles Löscht alle gespeicherten Profile, was manche Testfälle vereinfacht.

3.12.3 ProfileChangeListener

Dieses Interface wurde neu erstellt. Alle Screens die bei einem *Profile*-Wechsel informiert werden müssen dieses Interface implementieren und werden über die *onProfileChange* Methode benachrichtigt.

3.13 Package: game.sound

3.13.1 Neuerungen gegenüber dem Entwurf

Das Abspielen von Musik oder Soundeffekten war im Entwurf noch nicht berücksichtigt, deshalb wurde dieses Package mit all seinen Inhalten neu erstellt. Die zwei in diesem Package enthaltenen Klassen sind:

SoundHelper Die Aufgabe dieser Klasse ist es Sound und Musik aus den assets zu laden.

SoundController Diese Klasse enthält Funktionalitäten um Sounds und Musik abzuspielen.

3.14 Package: ui

Das gesamte Paket ui und die Unterpakete wurde wegen mangelnder Detailkenntnisse des benutzten Framesworks nur sehr allgemein entworfen. Trotzdem wurden hauptsächlich einige Hilfsklassen zu besseren Übersicht und Kapselung hinzugefügt.

3.14.1 ConfirmInterface

Hierbei handelt es sich um ein Interface, das lediglich die Klassen **yes()** und **no()** beinhaltet. Der Sinn ist es, eine Abstahierung für einfache zweiseitige Nutzerentscheidung zu bieten: Genutzt wird dieses Interface, um über einen Dialog (vgl: YesNoDialog) mitzuteilen, was bei Bestätigung und Ablehnung einer gestellten Option zu passieren hat.

3.14.2 Änderungen im StyleHelper

Der StyleHelper benutzt außer der "Skin" nun zusätzlich einen FreeTypeFontGenerator, um Schrift in allen beliebigen Größen darzustellen, da libgdx nur die Benutzung von Bitmap Fonts unterstützt. Er ist außerdem auf das Entwurfsmuster "Singleton" ausgelegt.

Neue Methoden

Methoden die nicht genauer erläutert sind, geben einfach den entsprechenden in der **skin.json** Datei definierten Style zurück.

- dispose() : Entfernt die genutzten Dateien aus dem Arbeitsspeicher.
- getButtonStyle()
- getTextButtonStyleLevel()
- getTextButtonStyleSquare()
- getImageButtonStyleRound()
- getImageButtonStyle(String icon) : Erstellt einen Style, der das per String definierte Icon als Bild besitzt.

- getImageButtonStyleRound(String icon) : Erstellt einen Style, der das per String definierte Icon als Bild besitzt.
- getImageTextButtonStyleTransparent()
- getImageTextButtonStyle(String icon) : Erstellt einen Style, der das per String definierte Icon als Bild besitzt.
- getImageTextButtonStyleTransparent(String icon) : Erstellt einen Style, der das per String definierte Icon als Bild besitzt.
- getBlackLabelStyle()
- getLabelStyle(int size) : Erstellt den Style mit der übergebenen Schriftgröße.
- getBlackLabelStyle(int size) : Erstellt den Style mit der übergebenen Schriftgröße.
- getCheckBoxStyle()
- getSliderStyle()
- getTextFieldStyle()
- getSelectBoxStyle()
- getDialogStyle()
- getWindowStyle()
- getDrawable(String path) : Gibt, falls vorhanden, die Grafik zurück, die im verwendeten Texturatlas durch den gegebenen String repräsentiert wird.

3.15 Package: ui.actors

Dieses Paket enthält jetzt einige Klassen mehr, die hauptsächlich der vereinfachten grafischen Darstellung dienen und deshalb von der Klasse “Actor“ von libgdx (oder deren Unterklassen) erben.

3.15.1 Neue Klassen

HintDialog Zeigt einen visuellen Hinweis und kann mit einem entsprechenden Button geschlossen werden. Wird im Platziermodus für die Hinweisanzeige genutzt.

IngameMenuDialog Enthält das feste Format des Spielmenüs, sowie sämtliche Listener (siehe Pflichtenheft 10.5.7 Spielmenü).

NewAchievementDialog Zeigt Beschreibung und Icon eines Achievements, sowie einen Button zum Schließen. Es kann auch eingestellt werden, dass zusätzlich eine Glückwunschnachricht angezeigt wird (siehe Pflichtenheft 10.5.9 Achievement-Benachrichtigung).

MaskedImage Kombiniert ein Bild mit einer beliebigen Maske, um z.B. Transparenzbereiche festzulegen.

NotificationDialog Ein Dialog, der eine beliebige Nachricht darstellt, sowie einen Button zum Schließen. Das Aussehen des Dialogs ist festgelegt.

PagedScrollPane Simuliert das für Geräte mit Toucheingabe übliche "Blättern" zwischen Ansichten, da libgdx keine solche Funktionalität bereitstellt. Es können verschiedene Widgets eingegeben werden, die dann nebeneinander dargestellt werden (es kann auch nur horizontal gescrollt werden). Durch eine "Swipe"-Geste kann der nächste Eintrag erreicht werden, der dann zentriert dargestellt wird.

ProfileButton Dies ist lediglich ein Button, der eigenständig aus einem Profil die nötigen Daten (im Moment Profilbild und Name) herausliest, und diese wie gewünscht darstellt.

YesNoDialog Ein Dialog, der eine Nachricht darstellt und dem Nutzer zwei Optionen bietet ("Ja" oder "Nein"). Es kann über ein ConfirmInterface (siehe package de.croggle.ui) festgelegt werden, was bei der Wahl der jeweiligen Option geschehen soll.

3.15.2 SonstigeÄnderungen

- Die **ObjectBar** wurde in das Paket de.croggle.ui.renderer verschoben, da sie dort intensiver genutzt wird.

3.16 Package: ui.renderer

Für das „renderer“ Paket wurde während der Entwurfsphase nur ein oberflächliches Konzept erstellt. Daher kam es zu umfangreichen Erweiterungen und teilweise auch zu Änderungen am Entwurf.

3.16.1 Änderungen gegenüber dem Entwurf

Die wichtigste Änderung am Entwurf betrifft die Vererbungshierarchie der BoardActors. Während der Entwurf noch einen ParentActor vorsah, empfahl es sich während der Implementierung eine gemeinsame Oberklasse für gefärbte BoardObjects zu haben. Dies liegt an der ähnlichen vorgehensweise beim Zeichnen der unterschiedlichen Farben. An die Stelle des ParentActors tritt somit der ColoredBoardObjectActor.

3.16.2 Neuerungen gegenüber dem Entwurf

Das Erstellen und die Synchronhaltung einer Repräsentation zu einem Board ist ein komplexes Thema, welches im Entwurf noch nicht detailliert beachtet wurde. Für den hierfür erforderlichen Code wurde daher nachträglich die mit dem Präfix ActorLayout versehenen Klassen angelegt. Diese werden hier kurz erläutert:

ActorLayout Kapselt die für die Darstellung benötigten Actors. Beinhaltet zudem Methoden für den Lookup BoardObject $\leftrightarrow$ BoardObjectActor, die Konfiguration, mit der es erstellt wurde und gecachete Statistiken zu dem Layout.

ActorLayouter Stark konfigurierbare, abstrakte Elternklasse, um die Daten eines Layouts zu generieren. Dies sind vor allem Positionen und Größen von BoardObjectActors. Durch das Anbieten von Schablonenmethoden und Callbacks kann der gleiche Code genutzt werden, um Layouts zu kreieren, als auch, um sie an Änderungen im Board anzupassen.

ActorLayoutConfiguration Kapselt alle Informationen, die der ActorLayouter zum Erstellen der Informationen benötigt. Dazu gehören z.B. Paddings, der vertikale Skalierungsfaktor etc.

ActorLayoutStatistics Ort, um Statistiken eines Layouts zu cachen, wie z.B. die Breitenkarte (width map), die jedem Actor die Breite des von ihm bewurzelten Unterbaumes zuweist.

ActorLayoutBuilder Helferklasse, um zu einem gegebenen Board ein Layout zu erstellen. Der Hauptteil des dafür benötigten Codes wird von ActorLayouter geerbt. Lediglich Schablonenmethoden zum Erzeugen von zu layoutenden BoardObjectActors werden implementiert.

ActorLayoutFixer Helferklasse, um eine Sammlung von ActorDeltas zu produzieren, die angewandt werden müssen, um ein gegebenes Layout an ein gegebenes Board anzupassen. Die Deltas können anschließend leicht animiert und umgesetzt werden (siehe hierzu BoardActorBoardChangeAnimator) Der Hauptteil des dafür benötigten Codes wird von ActorLayouter geerbt. In der Schablonenmethode zur Erzeugung von zu layoutenden BoardObjectActors werden immer wieder die gleichen Actors übergeben. Danach müssen in einer callback Methode lediglich Deltas zu den bereits im Layout vorhandenen Actors gebildet werden.

ActorDelta Fasst Änderungen an einem Actor zusammen. Diese werden vom BoardActorBoardChangeAnimator benutzt, um das Layout an den Zustand des Boards anzupassen.

Desweiteren ist anzumerken, dass der BoardActor zu einer hoch integrierten grafischen Komponente gewachsen ist. Zu seinem Funktionsumfang gehören:

BoardActorZoomAndPan Zooming und Panning Funktionalitäten für das dargestellte Layout

BoardActorLayoutEditing Boardbearbeitungsfunktionalitäten, wie z.B. Farbauswahl Popups für ungefärbte Actos oder Drag and Drop.

BoardActorBoardChangeAnimator Animationen bei eingehenden BoardEvents

Alle diese Funktionen sind jedoch gesondert in Module gekapselt ohne sich gegenseitig direkt zu referenzieren. Dazu bietet der BoardActor ein erweitertes Interface, welches aber nur im Paket „renderer“ zur Verfügung steht.

Zum Schluss seien noch die verbliebenen, nicht im Entwurf enthaltenen Klassen genannt und kurz erklärt:

BoardObjectActorDragging Helferklasse für alle Akteure, die bei Drag&Drop mit BoardObjectActors partizipieren möchten. Für diesen Zweck werden für alle drei BoardObjectActor Typen Actors erzeugt, die in libgdx' DragAndDrop als DragActor, ValidDragActor und InvalidDragActor gesetzt werden können.

ColorSelectorPopup Dedizierte grafische Komponente, mit der BoardActorLayoutEditing dem Nutzer Eingabemöglichkeiten zur Änderung von Farben von Colored-

BoardObjects bereitstellt.

ObjectBar Grafisches Element, welches eine Leiste darstellt, aus der neue BoardObjectActors mittels Drag&Drop gezogen und auf dem Spielfeld platziert werden können. Außerdem werden bereits vorhandene BoardObjectActors, die auf die Leiste gezogen werden, gelöscht.

PlaceHolderActor Eine Erweiterung des BoardObjectActors, welcher keine grafische Repräsentation bietet, aber dafür als Platzhalter dient. Werden vom Nutzer andere BoardObjectActors auf den Platzhalter geschoben, so wird die zugrundeligende Konstellation so modifiziert, dass der verschobene Actor den Platz des Platzhalters einnimmt.

TreeGrowth Enum (Aufzählung) von Richtungen, in die im Allgemeinen ‘"gewachsen“ werden kann. Wird hauptsächlich zur Angabe verwendet, in welche Richtung der Layout Baum wachsen soll.

WorldPane Paketsichtbare, grafische Komponente, mit der der BoardActor eine Übersetzung von Bildschirm- bzw. Szenenkoordinaten in die Koordinaten der gezoomten und verschobenen Layout-Welt realisiert.

3.17 Package: ui.screens

3.17.1 AbstractScreen

Hinzugefügte Methoden

getViewportWidth und getViewportHeight Getter für die Breite und Höhe des Viewports (des tatsächlich angezeigten Bereiches).

initializeWidgets In dieser Methode sollten alle Widgets, die im TableLayout der Ansicht verarbeitet werden initialisiert werden. Dass dies nicht einfach im Konstruktor geschieht, hat den Grund, dass das Laden aller Grafiken am Ende der AlligatorApp.create() Methode gebündelt wird, und erst danach Widgets erstellt werden können.

areWidgetsInitialized Um zu verhindern, dass initializeWidgets() bei jedem Anzeigen einer Ansicht neu gestartet wird, was auf die Performanz schlagen würde, signalisiert diese Methode, ob die Initialisierung schon erfolgt ist.

showLogicalPredecessor Zeigt den Screen, der auf einer logischen Ebene direkt über dem aktuell Angezeigten liegt. Per Default ist dies derselbe, der beim Drücken des "Back"-Buttons gezeigt würde. Diese Methode muss überschrieben werden, wenn es einen genauer definierten Vorgänger gibt.

3.17.2 AchievementScreen

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

Modifizierte Methoden

Konstruktor Nimmt nicht mehr den einen *GameController* als Argument entgegen.

3.17.3 CreditsScreen

Neuer Screen, der die Namen der App-Entwickler zeigt sowie einen Hinweis auf Bret Victor's Idee und seine Webseite.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

3.17.4 LevelPackagesScreen

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

showLogicalPredecessor Zeigt den Screen, der auf einer logischen Ebene direkt über dem aktuell Angezeigten liegt: in diesem Fall das Hauptmenü.

Modifizierte Methoden

Konstruktor Nimmt nicht mehr den einen *LevelPackageController* als Argument entgegen.

3.17.5 LevelsOverviewScreen

Hinzugefügte Methoden

showLogicalPredecessor Zeigt den Screen, der auf einer logischen Ebene direkt über dem aktuell Angezeigten liegt: in diesem Fall die Levelpaketübersicht.

3.17.6 LoadingScreen

Neuer Screen, der während Ladephasen des Spiels angezeigt wird. Er ersetzt außerdem den Splashscreen.

Hinzugefügte Methoden

render Zeigt den aktuellen Ladefortschritt an.

3.17.7 MainMenuScreen

Klasse implementiert jetzt noch zusätzlich *ProfileChangeListener*.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onProfileChange Methode wird aufgerufen, wenn das aktuelle Profil geändert wird.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.17.8 MultipleChoiceScreen

Klasse implementiert jetzt *SettingsChangeListener*.

Hinzugefügte Methoden

onShow Methode wird beim Zeigen des Screens aufgerufen.

hide Methode wird aufgerufen, wenn der Screen ausgeblendet wird.

render Methode wird zum Rendern des Screens aufgerufen.

onSettingsChange Methode wird aufgerufen wenn die Einstellungen geändert werden.

Modifizierte Methoden

Konstruktor Nimmt statt einem *GameController* einen *MultipleChoiceGameController* als Argument entgegen.

3.17.9 PlacementModeScreen

Klasse implementiert jetzt *SettingsChangeListener*.

Hinzugefügte Methoden

onShow Methode wird beim Zeigen des Screens aufgerufen.

hide Methode wird aufgerufen, wenn der Screen ausgeblendet wird.

render Methode wird zum Rendern des Screens aufgerufen.

onSettingsChange Methode wird aufgerufen wenn die Einstellungen geändert werden.

Modifizierte Methoden

Konstruktor Nimmt statt einem *GameController* einen *MultipleChoiceGameController* als Argument entgegen.

3.17.10 ProfileSetAvatarScreen

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onShow Methode wird beim Zeigen des Screens aufgerufen.

setProfileName Setter für den Profilnamen.

setIsInEditMode Teilt dem Screen mit, dass er zur Profilbearbeitung aufgerufen wurde, was z.B. bewirkt, dass er bei Bestätigung der Namenseingabe nicht zur Avatarauswahl schaltet, sondern das Profil entsprechend aktualisiert.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.17.11 ProfileSetNameScreen

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onShow Methode wird beim Zeigen des Screens aufgerufen.

setIsInEditMode Teilt dem Screen mit, dass er zur Profilbearbeitung aufgerufen wurde, was z.B. bewirkt, dass er bei Bestätigung kein neues Profil erstellt, sondern das aktuelle bearbeitet.

showBackButton Legt fest ob der Button, der zurück zum vorigen Screen navigiert, angezeigt wird oder nicht.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.17.12 QuitGameOverlay

Neues Overlay, dass beim Verlassen des Spiels den Spieler noch einmal fragt ob er wirklich das Spiel verlassen will.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onShow Methode wird beim Zeigen des Screens aufgerufen.

render Methode wird zum Rendern des Overlays aufgerufen.

3.17.13 SelectProfileScreen

Klasse implementiert jetzt ProfileChangeListener.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onProfileChange Methode wird aufgerufen wenn das aktuelle Profil geändert wird.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.17.14 SettingsScreen

Klasse implementiert jetzt ProfileChangeListener.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onProfileChange Methode wird aufgerufen wenn das aktuelle Profil geändert wird.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.17.15 SimulationModeScreen

Klasse implementiert jetzt SettingsChangeListener.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onProfileChange Methode wird aufgerufen wenn das aktuelle Profil geändert wird.

render Zeichnet alle erforderlichen Änderungen

onShow Methode wird beim Zeigen des Screens aufgerufen.

hide Methode wird aufgerufen wenn der Screen ausgeblendet wird.

onSettingsChange Methode wird aufgerufen wenn die Einstellungen geändert werden.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.17.16 SplashScreen

Funktionalität wurde mit den LoadingScreen zusammengeführt. Die Klasse selbst wurde entfernt.

3.17.17 StatisticScreen

Klasse implementiert jetzt ProfileChangeListener.

Hinzugefügte Methoden

initializeWidgets Initialisierung von Widgets.

onProfileChange Methode wird aufgerufen wenn das aktuelle Profil geändert wird.

onShow Methode wird beim Zeigen des Screens aufgerufen.

Modifizierte Methoden

Konstruktor Nimmt nur noch eine *AlligatorApp* als Argument entgegen.

3.18 Package: util

3.18.1 PatternBuilder

Neu hinzugefügte Klasse zur Erstellung verschiedener Muster für die Verwendung im Farbenblindmodus.

Hinzugefügte Methoden

generateEmpty Generiert ein leeres Muster.

generateCircle Generiert ein Muster aus ausgefüllten Kreisen.

generateVerticalLines Generiert ein Muster aus vertikalen Streifen.

generateHorizontalLines Generiert ein Muster aus horizontalen Streifen.

generateCheckerboard Generiert ein Schachbrettmuster.

generateRhombus Generiert ein Muster aus ausgefüllten Rauten.

generateFilled Generiert ein ausgefülltes Muster.

generateTriangleStrip Generiert ein Muster aus Dreiecken.

3.18.2 RingBuffer

Hinzugefügte Methoden

size Methode zum Ermitteln der Anzahl an Elementen im *RingBuffer*.

3.19 Package: util.convert

Das Paket „convert“ wurde erst in der Implementierung eingeführt. Es beinhaltet statische Funktionen, um Objekte zu konvertieren. Hauptsächlich werden dabei verschiedene externe Formate und die applikationsinterne Darstellung von λ-Termen ineinander überführt. Diese externen Formate sind die gebräuchliche λ-Notation und JSON, wie es für die Levelspezifikation beschrieben ist.

AlligatorToJson Methode überträgt applikationsinterne Konstellationen (BoardObjects) in das spezifizierte JSON Format

AlligatorToLambda Methode überträgt applikationsinterne Konstellationen (BoardObjects) in die gebräuchliche λ-Notation

JsonToAlligator Methode übersetzt nach der Spezifikation formatiertes JSON in Applikationsinterne Konstellationen (BoardObjects)

LambdaToAlligator Methode übersetzt Zeichenketten, die nach der gebräuchlichen λ-Notation formatiert sind, in Applikationsinterne Konstellationen (BoardObjects)

Desweiteren können in der Klasse „ColorConvert“ Strings, die die aus HTML bekannte Farbkodierung beinhalten, in das von libgdx verwendete Farbformat übertragen werden.

4 Zeitlicher Ablauf

4.1 Lukas

28.12. - 09.01. Implementierung der Pakete *board*, *board.operations*, *board.operations.validation*, *util*, *data*.

09.01. - 17.01. Implementierung des Layouting.

17.01. - 29.01. Implementierung von *BoardActor* (renderer): Rendering, Zooming, Panning.

29.01. - 04.02. Bearbeiten des *Board* (Farbauswahl).

04.02. - 08.02. Implementierung von Drag and Drop.

08.02. - 10.02. Fehlerbehebung

4.2 Tobias

08.01. - 15.01. Achievementklassen und dazugehörige Tests fertiggestellt, AchievementController rudimentär implementiert

15.01. - 21.01. Tests für Achievementklassen verbessert, Achievementerstellung in AchievementFactory ausgelagert, Interaktion mit der Datenbank, Lokalisierung

21.01. - 30.01. AchievementScreen implementiert, Embleme für Achievement, Aufspalten der Embleme, Update Testcases für Achievementklassen, Test für AchievementController verbessert, Icons für alle TimeAchievements

30.01. - 04.02. Mehr Icons für verschiedene Achievements, Verbesserungen im AchievementScreen, mehr Localization und Verbesserungen an einigen bereits übersetzten

Stellen, Skalierung von allen bisherigen Achievementicons, Details im SimulationScreen

04.02. - 08.02. Berechnung der Spielzeit, Tests anpassen an Veränderungen im Code, mehrIcons für Achievements, ästhetische Details

4.3 Jonas

07.01. - 22.01. Implementation von *data.persistence.manager*.

13.01. - 26.01. Implementation von *game.profile*.

13.01. - 28.01. Implementation von *data.persistence*.

22.01. - 03.02. Einbinden der Controller in die Screens.

25.01. - 26.02. StatisticScreen implementiert.

01.02. - 02.02. Implementierung von *game.sound*.

30.01. - 08.02 Bugfixing und kleine Änderungen an den Screens.

4.4 Iris

13.01. - 20.01. Grundlegende Strukturen und einige grobe Layouts (skin.json, AbstractScreen, StyleHelper).

20.01. - 27.01. Mehr Layout, erste Listener für Navigation in der App, ein paar Dialoge, Grafiken für Buttons, Icons etc.

27.01. - 03.02. Mehr Grafiken, Einbauen von fertigem Inhalt, Layoutkorrekturen.

03.02. - 10.02. LoadingScreen, Steuerung des Ladeverhaltens, danach: Tutorials.

4.5 Vincent

24.12. Implementierung des Pakets *game.board*.

08.01. Hinzufügen von Tests für das Paket *util*.

10.01. Hinzufügen von Tests für *game.board*.

12.01. - 18.01. Implementierung fehlender Klassen in *game.board.operations* und entsprechender Tests.

18.01. - 25.01. Implementierung des Pakets *game* und entsprechender Tests.

27.01. - 28.01. Integration des *GameController* und *Simulator* in das User Interface.

29.01. - 02.02. Hinzufügen von Mustern für den Farbenblindmodus.

02.02. - 08.02. Fehlerbehebung in *GameController* und *Simulator*.

4.6 Lena

07.01. - 23.01. Implementation von game.level

18.01. - 20.01. Schreiben der Level Json des 1 Pakets

23.01. - 27.01. Implementation des MultipleChoiceScreen

25.01. - 30.01. Schreiben der Level Json des 2 Pakets

27.01. - 06.02. Hinzufügen von Achievement icons

04.02. Hinzufügen des Goal Dialogs

04.02. - 08.02. Hinzufügen der Hints und des HintDialogs

5 Neues Levelformat

Gegenüber dem Entwurf ergaben sich einige Änderungen an dem Format der Levelbeschreibung. Das neue, angepasste Format ist daher an dieser Stelle erneut dokumentiert.

5.1 Multiple-Choice-Level

"de.croggle" : Object - Namensraum
- *"levels"* : List - Liste mit Levelobjekten. Gewöhnlich nur ein Paket
  - : Object - Darstellung einzelner Levels
    - *"type"* : "multiple choice"
    - *"description"* : String - Beschreibung zu einem Level
    - *"design"* : String - Pfad zu einem Leveldesign
    - *"animation"* : List - Liste mit Pfaden zu Bildern die als Tutorial dienen
    - *"abort simulation after"* : Integer - Nach wie vielen Schritten ein Level gewonnen (positiv) oder verloren (negativ) ist, null wird für die Standardeinstellungen eingetragen
    - *"hints"* : List - Strings mit Pfaden zu Hilfegrafiken
    - *"data"* (multiple choice) : Object - Spezielle Daten für die einzelnen Leveltypen
      - *"initial"* : Object - Board Objekt der Anfangskonstellation
      - *"answers"* : List - Liste mit Board Objekten der möglichen Antworten
      - *"correct answer"* : unsigned Integer - Der Index der richtigen Antwort zur Fragestellung

5.2 Färbe-Level

"de.croggle" : Object - Namensraum
- *"levels"* : List - Liste mit Levelobjekten. Gewöhnlich nur ein Paket
  - : Object - Darstellung einzelner Levels
    - *"type"* : "color edit"
    - *"description"* : String - Beschreibung zu einem Level

- *"design"* : String - Pfad zu einem Leveldesign
- *"animation"* :List - Liste mit Pfaden zu Bildern die als Tutorial dienen
- *"abort simulation after"* : Integer - Nach wie vielen Schritten ein Level gewonnen (positiv) oder verloren (negativ) ist, null wird für die Standardeinstellungen eingetragen
- *"hints"* : List - Strings mit Pfaden zu Hilfegrafiken
- *"data"* (multiple choice) : Object - Spezielle Daten für die einzelnen Leveltypen
  - *"initial constellation"* : Object - Board Objekt der Anfangskonstellation
  - *"objective"* : Object - Board Objekt der zu erreichenden Konstellation
  - *"user colors"* : List - Liste mit 6 verschiedenen Integer Werten, die die Farben beschreiben, die der Benutzer zum Färben von Elementen benutzen soll
  - *"blocked colors"* : List - Liste mit Integer Werten, die blockierte Farben beschreiben

5.3 Einfüge-Level

"de.croggle" : Object - Namensraum
- *"levels"* : List - Liste mit Levelobjekten. Gewöhnlich nur ein Paket
  - : Object - Darstellung einzelner Levels
    - *"type"* : "term edit"
    - *"description"* : String - Beschreibung zu einem Level
    - *"design"* : String - Pfad zu einem Leveldesign
    - *"animation"* : List - Liste mit Pfaden zu Bildern die als Tutorial dienen
    - *"abort simulation after"* : Integer - Nach wie vielen Schritten ein Level gewonnen (positiv) oder verloren (negativ) ist, null wird für die Standardeinstellungen eingetragen
    - *"hints"* : List - Strings mit Pfaden zu Hilfegrafiken
    - *"data"* (multiple choice) : Object - Spezielle Daten für die einzelnen Leveltypen
      - *"initial constellation"* : Object - Board Objekt der Anfangskonstellation
      - *"objective"* : Object - Board Objekt der zu erreichenden Konstellation
      - *"user colors"* : List - Liste mit 6 verschiedenen Integer Werten, die die Farben beschreiben, die der Benutzer zum Färben von Elementen benutzen soll

- *"blocked colors"* : List - Liste mit Integer Werten, die blockierte Farben beschreiben
- *"blocked types"* : List - String Liste mit Namen von Elementtypen, die im Level nicht platzierbar sind (egg, colored alligator, aged alligator)

6 Unit Tests

6.1 Package: data

6.1.1 AndroidLocalizationBackendTest

testAppName Testet, ob der Applikationsname richtig bezogen wird

testPluralMissing Testet, ob das Backend richtig mit fehlenden Pluralen umgehen kann. Erwartet den eingegeben „key“ als Resultat.

testSetGetLocale Testet, ob die System-Locale richtig gesetzt und wieder ausgelesen werden kann.

testSystemLocale Testet, ob die default System-Locale zu beginn gesetzt ist.

6.1.2 TestLocalizationBackend

Helferklasse, um ohne ein konkretes, plattformabhängiges Backend den LocalizationHelper zu initialisieren und zu benutzen.

6.2 Package: data.persistence.manager

6.2.1 AchievementManager

testInsertUnlockedAchievements Dieser Test überprüft ob die freigeschalteten Achievents erfolgreich in der SQLite-Datenbank eingefügt werden.

testFetchUnlockedAchievements Dieser Test überprüft ob die Achievements nach dem Einfügen in die Datenbank auch wieder erfolgreich und mit unveränderten Werten

aus der Datenbank geladen werden können.

testInsertUnlockedAchievements Dieser Test überprüft ob in der Datenbank gespeicherte Achievements verändert werden können.

testDeleteUnlockedAchievments Dieser Test überprüft ob nach dem Löschen eines Benutzers auch die von ihm freigeschalteten Achievements automatisch aus der Datenbank gelöscht werden.

6.2.2 LevelProgressManager

testInsertLevelProgress Dieser Test überprüft ob die vom Benutzer erzielten Levelfortschritte erfolgreich in der SQLite-Datenbank eingefügt werden.

testFetchLevelProgress Dieser Test überprüft ob die Levelfortschritte nach dem Einfügen in die Datenbank auch wieder erfolgreich und mit unveränderten Werten aus der Datenbank geladen werden können.

testInsertLevelProgress Dieser Test überprüft ob in der Datenbank gespeicherte Levelfortschritte verändert werden können.

testDeleteLevelProgress Dieser Test überprüft ob nach dem Löschen eines Benutzers auch die von ihm erzielten Levelfortschritte automatisch aus der Datenbank gelöscht werden.

6.2.3 ProfileManager

testInsertProfile Dieser Test überprüft ob Benutzerprofile erfolgreich in der SQLite-Datenbank eingefügt werden.

testFetchProfile Dieser Test überprüft ob die Benutzerprofile nach dem Einfügen in die Datenbank auch wieder erfolgreich und mit unveränderten Werten aus der Datenbank geladen werden können.

testInsertProfile Dieser Test überprüft ob in der Datenbank gespeicherte Benutzerprofile verändert werden können.

testDeleteProfile Dieser Test überprüft ob die in der Datenbank gespeicherten Benutzerprofile auch wieder gelöscht werden können.

testDeleteProfile Dieser Test überprüft ob die in der Datenbank gespeicherten Benutzerprofile auch wieder gelöscht werden können.

testFetchAllProfiles Dieser Test überprüft ob alle in der Datenbank gespeicherten Benutzerprofile auch wieder auf einmal geladen werden können.

6.2.4 PersistenceManager

Probleme Diese Test können noch nicht erfolgreich ausgeführt werden, da es Probleme mit bei der Initialisierung bestimmter libgdx Objekte gibt.

6.2.5 SettingManager

testInsertLevelSetting Dieser Test überprüft die zum jeweiligen Benutzer gehörende Einstellung erfolgreich in der SQLite-Datenbank eingefügt werden.

testFetchLevelSetting Dieser Test überprüft ob die Einstellung nach dem Einfügen in die Datenbank auch wieder erfolgreich und mit unveränderten Werten aus der Datenbank geladen werden können.

testInsertSetting Dieser Test überprüft ob in der Datenbank gespeicherte Einstellungen verändert werden können.

testDeleteSetting Dieser Test überprüft ob nach dem Löschen eines Benutzers auch die zu ihm gehörende Einstellung automatisch aus der Datenbank gelöscht wird.

6.2.6 StatisticManager

testInsertLevelSetting Dieser Test überprüft die zum jeweiligen Benutzer gehörende Statistiken erfolgreich in der SQLite-Datenbank eingefügt werden.

testFetchLevelSetting Dieser Test überprüft ob die Statistiken nach dem Einfügen in die Datenbank auch wieder erfolgreich und mit unveränderten Werten aus der Datenbank geladen werden können.

testInsertSetting Dieser Test überprüft ob in der Datenbank gespeicherte Statistiken verändert werden können.

testDeleteSetting Dieser Test überprüft ob nach dem Löschen eines Benutzers auch die

zu ihm gehörende Statistik automatisch aus der Datenbank gelöscht wird.

6.3 Package: game

6.3.1 ColorTest

testMaxColorsEqualsColorStringLength Testet, ob genug Farben vorhanden sind, um die maximale Anzahl an möglichen Farben abzudecken.

6.3.2 SimulatorTest

testOmega Testet die korrekte Auswertung eines Schrittes des Ω-Terms.

testLevel2 Testet die korrekte Auswertung des zweiten Levels aus dem Entwurf.

testLevel8 Testet die korrekte Auswertung des achten Levels aus dem Entwurf.

testLevel10 Testet die korrekte Auswertung des zehnten Levels aus dem Entwurf.

testLevel12 Testet die korrekte Auswertung des zwölften Levels aus dem Entwurf.

testTwoAlligators Testet die korrekte Auswertung eines Terms mit zwei Alligatoren.

testTakeFirst Testet die korrekte Auswertung eines Terms, bei dem das erste von zwei Eiern übrig bleibt.

testTakeSecond Testet die korrekte Auswertung eines Terms, bei dem das zweite von zwei Eiern übrig bleibt.

testOldAlligator Testet die korrekte Auswertung eines Terms, der einen alten Alligator enthält.

testColorRule Testet die korrekte Auswertung eines Terms, bei dem eine Umfärbung notwendig ist.

testYCombinatorOneStep Testet die korrekte Auswertung eines Schrittes des Y-Combinators.

testIncrementZero Testet die korrekte Auswertung einer Inkrementierung von 0 mit Church-Numeralen.

testOnePlusOne Testet die korrekte Auswertung von $1 + 1$ mit Church-Numeralen.

testThreePlusFour Testet die korrekte Auswertung von $3 + 4$ mit Church-Numeralen.

6.4 Package: game.achievement

6.4.1 AchievementControllerTest

testUpdateAchievements Dieser Test überprüft die Interaktion der updateAchievements - Methode mit den verschiedenen AchievementListen und ihren Inhalten für unterschiedliche Statistiken als Eingabe.

testConvert Dieser Test überprüft das Konvertieren von einem SparseIntArray, so wie den der später als Input von der Datenbank kommt, in eine Liste von Achievements mit entsprechenden Indices.

6.4.2 AlligatorsEatenAchievementTest

testInitialize Dieser Test überprüft, ob das AlligatorsEatenAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.4.3 AlligatorsEatenPerLevelAchievementTest

testInitialize Dieser Test überprüft, ob das AlligatorsEatenPerLevelAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.4.4 AlligatorsPlacedAchievementTest

testInitialize Dieser Test überprüft, ob das AlligatorsPlacedAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.4.5 AlligatorsPlacedPerLevelAchievementTest

testInitialize Dieser Test überprüft, ob das AlligatorsPlacedPerLevelAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.4.6 HintPerLevelAchievementTest

testInitialize Dieser Test überprüft, ob das HintPerLevelAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.4.7 LevelAchievementTest

testInitialize Dieser Test überprüft, ob das LevelAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.4.8 TimeAchievementTest

testInitialize Dieser Test überprüft, ob das TimeAchievement nach dem Ausführen der initialize-Methode in den Attributen die richtigen Werte stehen hat.

testRequirementsMet Dieser Test überprüft, ob die requirementsMet-Methode den Index des Achievements auf die richtige Art und Weise manipuliert.

6.5 Package: game.board

6.5.1 AlligatorTest

testParent Testet das Setzen eines Elternalligators.

testIsMovable Testet die richtige Interpretation der *movable* Flags bei der Erstellung eines neuen Alligators.

testIsRemovable Testet die richtige Interpretation der *removable* Flags bei der Erstellung eines neuen Alligators.

6.5.2 ColoredBoardObjectTest

testGetSetColor Testet das Setzen einer Farbe für alle implementierenden Klassen.

testIsRecolorable Testet die richtige Interpretation der *recolorable* Flags für alle implementierenden Klassen.

6.5.3 ParentTest

testAddChild Testet das Hinzufügen eines Kindelements zu einem *Parent*-Objekt am Ende.

testInsertChild Testet das Hinzufügen eines Kindelements zu einem *Parent*-Objekt an verschiedenen Stellen.

testRemoveChild Testet das Entfernen eines Kindelements von einem *Parent*-Objekt.

testReplaceChild Testet das Ersetzen eines Kindelements durch ein anderes in einem *Parent*-Objekt.

6.6 Package: game.board.operations

6.6.1 CollectBoundColorsTest

testSimpleBoundColor Überprüft an einem einfachen Beispiel, ob die Anzahl der Farben im *Board* richtig berechnet wird und die richtigen Farben erkannt werden.

testRootBoundColor Überprüft ob erkannt wird, dass das *Board* selbst keine Farbe hat.

testMultipleBoundColor Überprüft, ob auch bei einem *Board* mit mehreren Farben die Anzahl und Art der Farben richtig erkannt wird.

testBoundColorsMultipleOccurences Überprüft, ob Anzahl und Farbe der Farben richtig erkannt wird, wenn mehrere Farben mehrmals im *Board* vorkommen.

6.6.2 CollectFreeColorsTest

testSimpleFreeColor Überprüft die Erkennung eines freistehenden gefärbten Eis und dessen Farbe.

testSimpleNonFreeColor Überprüft ob nicht alleine stehende Eier als solche erkannt werden.

testRootFreeColor Überprüft, dass das *Board* nicht als alleine stehendes Ei erkannt wird.

testFreeColorWithParent Überprüft die Erkennung eines anders als sein Parent gefärbten Eis und dessen Farbe.

testMultipleFreeColors Überprüft die Erkennung von mehreren anders als die jeweiligen Parents gefärbten Eiern und deren Farbe.

testFreeColorMultipleOccurences Überprüft die Erkennung von mehreren anders als die jeweiligen Parents gefärbten Eiern und deren Farbe, auch wenn diese mehrmals auftritt.

6.6.3 CreateDepthMapTest

testSimple Überprüft an einem einfachen Beispiel ob für jedes *BoardObject* erkannt wird in welcher Ebene des Baumes es liegt.

6.6.4 CreateHeightMapTest

testSimple Überprüft an einem einfachen Beispiel ob für jedes *BoardObject* erkannt wird wie hoch es ist.

testCase0 Überprüft ob die Höhe eines *Boards* richtig berechnet wird.

6.6.5 CreateWidthMapTest

testSimple Überprüft an einem einfachen Beispiel ob für jedes *BoardObject* erkannt wird wie breit es ist.

testCase0 Überprüft ob die Breite eines *Boards* richtig berechnet wird.

6.6.6 ExchangeColorTest

testSimple Überprüft an einem einfachen Beispiel ob das umfärben *BoardObjects* richtig funktioniert.

6.6.7 FindEatingTest

testSimple Überprüft an einem einfachen Beispiel ob das Finden des *Alligators*, der als nächstes fressen kann, funktioniert.

testPrecedence Überprüft ob der richtige *Alligator*, als derjenige der als nächstes fressen kann, zurückgegeben wird an einem komplizierterem Beispiel.

6.6.8 FlattenTreeTest

testSimple Überprüft die Kompirimierung des *Boards* als Liste.

6.6.9 GetParentHierachyTest

testSimple Überprüft an einem einfachen Beispiel ob für ein Kind die richtige Parent Hierachie zurückgegeben wird.

6.6.10 ReplaceEggsTest

testSimple Überprüft an einem einfachen Beispiel ob das Ersetzten der Eier eines Alligators korrekt funktioniert.

testSimpleRecolorFree Überprüft an einem einfachen Beispiel ob das Ersetzten der Eier eines Alligators korrekt funktioniert, wenn keine Umfärbungen nötig sind.

testSimpleRecolorBound Überprüft an einem einfachen Beispiel ob das Ersetzten der Eier eines Alligators korrekt funktioniert, wenn Umfärbungen nötig sind.

6.7 Package: game.board.operations.validation

6.7.1 FindBoardErrorsTest

testUncoloredEgg Testet, ob ein ungefärbtes Ei auf einem *Board* auch als ein solches erkannt und behandelt wird und eine dementsprechende Warnung auftritt.

testUncoloredalligator Testet, ob ein ungefärbter *ColoredAlligator* auf einem *Board* auch als ein solcher erkannt und behandelt wird und eine dementsprechende Warnung auftritt.

testEmptyAgedAlligator Testet, ob ein *AgedAlligator* ohne Kinder auf einem *Board* auch als ein solcher erkannt und behandelt wird und eine dementsprechende Warnung auftritt.

testEmptyBoard Testet, ob ein leeres *Board* auch als ein solches erkannt und behandelt wird.

testEmptyColoredAlligator Testet, ob ein *ColoredAlligator* ohne Kinder auf einem *Board* auch als ein solcher erkannt und behandelt wird und eine dementsprechende Warnung auftritt.

testNoUncolored Testet, ob ein gültiges *Board* mit eingefärbten Objekten auch als gültig erkannt wird und eine dementsprechende Warnung auftritt.

testNoEmptyAgedAlligator Testet, ob ein gültiges *Board* mit einem *AgedAlligator* mit Kindern auch als gültig erkannt wird und eine dementsprechende Warnung auftritt.

testNoEmptyBoard Testet, ob ein nicht leeres gültiges *Board* auch als ein solches erkannt und behandelt wird und eine dementsprechende Warnung auftritt.

testEmptyNoColoredAlligator Testet, ob ein *ColoredAlligator* mit Kindern auf einem *Board* auch als gültig erkannt und behandelt wird und eine dementsprechende Warnung auftritt.

testMultipleErrors Testet ob die richtge Anzahl an Warnungen gegeben wird, sollten mehrere Fehler auftreten, die ein *Board* ungültig werden lassen.

6.7.2 ValidateConstellationTest

testUncoloredEgg Testet, ob ein ungefärbtes Ei auf einem *Board* auch als ein solches erkannt und behandelt wird.

testUncoloredalligator Testet, ob ein ungefärbter *ColoredAlligator* auf einem *Board* auch als ein solcher erkannt und behandelt wird.

testEmptyAgedAlligator Testet, ob ein *AgedAlligator* ohne Kinder auf einem *Board* auch als ein solcher erkannt und behandelt wird.

testEmptyBoard Testet, ob ein leeres *Board* auch als ein solches erkannt und behandelt wird.

testEmptyColoredAlligator Testet, ob ein *ColoredAlligator* ohne Kinder auf einem *Board* auch als ein solcher erkannt und behandelt wird.

testNoUncolored Testet, ob ein gültiges *Board* mit eingefärbten Objekten auch als gültig erkannt wird.

testNoEmptyAgedAlligator Testet, ob ein gültiges *Board* mit einem *AgedAlligator* mit Kindern auch als gültig erkannt wird.

testNoEmptyBoard Testet, ob ein nicht leeres gültiges *Board* auch als ein solches erkannt und behandelt wird.

testEmptyNoColoredAlligator Testet, ob ein *ColoredAlligator* mit Kindern auf einem *Board* auch als gültig erkannt und behandelt wird.

6.8 Package: game.level

6.8.1 LevelControllerTest

testSize Testet ob der *LevelController* die richtige Anzahl an Leveln lädt.

testLevel Testet ob das erste Level der Liste auch das erste Level ist.

6.8.2 LeveLoadHelperTest

testCase0 Überprüft das Json von Level 00x00.

testCase1 Überprüft das Json von Level 00x01.

testCase2 Überprüft das Json von Level 00x02.

testCase3 Überprüft das Json von Level 00x03.

testCase4 Überprüft das Json von Level 00x04.

testCase5 Überprüft das Json von Level 00x05.

testCase6 Überprüft das Json von Level 00x06.

testCase7 Überprüft das Json von Level 00x07.

testCase8 Überprüft das Json von Level 00x08.

testCase9 Überprüft das Json von Level 00x09.

testCase10 Überprüft das Json von Level 00x10.

testCase11 Überprüft das Json von Level 00x11.

testCase20 Überprüft das Json von Level 01x00.

testCase21 Überprüft das Json von Level 01x01.

testCase22 Überprüft das Json von Level 01x02.

6.8.3 LoadColorEditLevelFromJsonTest

testCase0 Überprüft ob das Json von Level 00x00 richtig ausgelesen wird.

testCase1 Überprüft ob das Json von Level 00x01 richtig ausgelesen wird.

testCase5 Überprüft ob das Json von Level 00x05 richtig ausgelesen wird.

testCase8 Überprüft ob das Json von Level 00x08 richtig ausgelesen wird.

6.8.4 LoadPackageTest

testLoading Testet ob der *LevelPackagesController* die richtige Anzahl an Levelpaketen lädt.

testLoadedValues Testet ob die Werte korrekt aus den Json Dateien ausgelesen werden.

6.9 Package: util

6.9.1 RingBufferTest

testPush Testet das Hinzufügen eines Elements.

testEmptyPop Testet das richtige Verhalten beim Versuch ein Element zu entfernen, obwohl keines vorhanden ist.

testMinSize Testet, ob mindestens die vorgegebene Anzahl an Elementen gespeichert werden kann.

testMazSize Testet, ob höchstens die vorgegebene Anzahl an Elementen gespeichert wird.

6.10 Package: util.convert

6.10.1 AlligatorToJsonTest

testSimpleManual Testet ob ein *Board* richtig in einen äquivalenten Json-Ausdruck umgewandelt wird.

6.10.2 AlligatorToLambdaTest

testSimpleManual Testet ob ein *Board* richtig in einen äquivalenten λ-Term umgewandelt wird.

6.10.3 JsonToAlligatorTest

testSimpleManual Testet ob ein Json-Ausdruck richtig in ein äquivalente *Board* umgewandelt wird.

6.10.4 LambdaToAlligatorTest

testSimpleManual Testet ob ein λ-Term richtig in ein äquivalente *Board* umgewandelt wird.