TextAdventure Dokumentation

Aufbau

Das Projekt ist in Kotlin geschrieben und die einzig benötigte Bibliothek ist Swing, ein Java-UI-Framework, welches aufgrund der Kompatibilität zwischen Kotlin und Java genutzt werden kann.

Das Projekt ist in Model, View und Controller unterteilt. Model, View und Controller sind separiert und befinden sich in ihren jeweiligen packages.

Controller

Hier befindet sich die Hauptklasse Controller und eine Combat-Klasse.

Die Controller Klasse beinhaltet Methoden für Kommunikation zwischen Model und View im Fall von komplizierteren Situationen wie Formatieren von Informationen für den View und aktualisieren von Informationen, deren Komplexität über die von Swing bereitgestellten Listeners zwischen Model und View hinausgehen. Außerdem beinhaltet sie die parseInput-Methode, welche für das Verarbeiten von Spielyeingaben außerhalb von Kämpfen benutzt wird.

Combat

Die Combat-Klasse implementiert eine Zustandsmaschine für den rundenbasierten Kampf zwischen Held und Gegner. Im Normalmodus (mode 0) werden Standardbefehle wie “attack”, “defend”, “use” und “escape” verarbeitet. Wird ein kritischer Treffer ausgelöst, wechselt die Klasse in den Puzzle-Modus (mode 1), in dem ein zufällig generiertes Puzzle vom Spiely korrekt und schnell eingegeben werden muss – die Eingabe wird mittels Needleman-Wunsch-Algorithmus und Zeitmessung ausgewertet, um den Schadensmultiplikator zu bestimmen. Die Methode combatParse() gibt unterschiedliche Rückgabecodes (0: Kampf wird fortgesetzt, 1: Gegner besiegt, 2: Held gestorben, 3: erfolgreiche Flucht) zurück, die der Controller nutzt, um den Spielzustand (z.B. Aktualisierung von Map und Info oder Verarbeitung des Heldentods) anzupassen und den Kampf gegebenenfalls zu beenden.

Model

Das Model hält alle für das Spiel benötigte Daten, wie die Karte, den Helden und alle Gegenstände und Gegner. Das package ist in subpackages für Funktionsabschnitte unterteilt, das Hauptpackage enthält die Klasse Model, welche alle benötigten Teile des Models enthält.

Des Weiteren beinhaltet das Hauptpackage die Klasse Map, in der während der Initialisierung die gesamte prozedurale Generation der Karte durchgeführt wird, inklusive Gegner-, Raum- und Itemplatzierung. Die Map-Klasse beinhaltet auch Methoden zum Bewegen des Spielys (bzw. von Entities generell) und zum Berechnen von Nachbarräumen zu Räumen (als Koordinaten oder als Room-Objekte).

Das Objekt Data beinhaltet Blaupausen für Gegner und Gegenstände, welche während der Map-Initialisierung kopiert und je nach aktuellem floor angepasst werden.

world

Im subpackage world befinden sich Klassen, die generell mit der Karte zusammenhängen. Eine davon ist Room, sie repräsentiert einen Raum in der Karte und hat ein Inventory um Gegenstände zu speichern sowie eine Liste aus Entitäten, um die im Raum befindlichen Gegner und eventuell den Held speichern zu können. Außerdem befinden sich hier auch Exceptions, welche im Fall von nicht validen Aktionen genutzt werden. Das Objekt Directions dient zum korrelieren von lesbaren Himmerlsrichtungen zu intern genutzten Nummern.

base
Hier befinden sich alle Klassen, die unabhängig Kontexts der Karte benutzt werden können. Im Hauptpackage ist die Klasse Inventory, welche sowohl von Räumen als auch Entities zum Halten von Gegenständen genutzt wird. Sie erweitert die Klasse ArrayList, da sie viele der Eigenschaften teilt und überschreibt die Methoden add und remove, um eigene Exceptions nutzen und die maximale Größe beachten zu können.

entities
Klassen, die den Helden sowie die Gegner darstellen. Sowohl Held als auch Gegner erben von der Klasse Entity, damit sie praktisch in einer Liste gespeichert werden können, sind allerdings größtenteils identisch zur Klasse Entity, da dies eventuelle KI-Erweiterungen der Gegner vereinfacht hätte (z. B. verfolgen durch Räume).

Entity beinhaltet Methoden zum interagieren mit anderen Entities, sowie dem aktuellen Raum, außerdem Wrapper zum Ändern mancher Eigenschaften der Entity.

item
Alle Gegenstandsklassen. Alle hier liegenden Klassen erben von der abstraken Klassen Item, um, wie bei Entities, in einer Liste, dem Inventory gespeichert werden zu können. So erweitert z.B. die Klasse Armor die Item Klasse um die Absorbtion und die Anzahl der Aktionspunkte.

View
Der View enthält alle für den Endnutzer sichtbaren Elemente. Im Hauptpackage befinden sich die Hauptklasse View und die Klasse für die Menüleiste, MenuBar. Alle Klassen im View erben von Swing-Klassen, damit die Objektstruktur der View-Klassen die Struktur des Swing-UIs repräsentiert.

View
Klasse, die das eigentliche Spielfenster repräsentiert. Sie initialisiert die Menüleiste und den Conent (alles im Fenster, was nicht die Menüleiste ist) und setzt die ursprüngliche Größe des Fensters.

MenuBar
Die Menüleiste, enthält ein Menü namens File, welches als einzigen Unterpunkt ein Schließen des Spiels enthält. Diese Klasse sollte ursprünglich mehr Optionen enthalten.

Content
Der eigentliche Inhalt des Spielfensters. Dieser enthält den Spielyinput, den Output des Spiels und die Sidebar, welche Informationen über das Spiel enthält.

Sidebar
Diese Klasse enthält die Karte sowie einen Informationsabschnitt, beide repräsentiert durch eine Tabelle ohne Trennlinien. Die Informationstabelle wird mittels eines TableModels initialisiert, um spätere Änderungen einfacher zu machen. Zudem wird hier die Breite der Sidebar festgelegt, basierend auf einer festgelegten Zellengröße der Karte, da diese sonst standardmäßig die Hälfte des Fensters einnehmen würde.

UML-Diagramm