Raphael Zender et al. (Hrsg.): Die 18. Fachtagung Bildungstechnologien (DELFI),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2020 229

VITMaze – Die Java Coding-Challenge für Verwaltungsinformatiker

Patrick Stalljohann[1], Maik Merten[1]

Abstract: Die Motivation von Studierenden zur aktiven Beschäftigung mit der objektorientierten Programmierung bei sehr heterogenen Vorkenntnissen in interdisziplinären Studiengängen ist eine Herausforderung. Verfügbare Online-Programmierspiele adressieren dies insbesondere für den privaten Bereich, vernachlässigen aber häufig Aspekte der Code-Qualität, der Modularisierung und der Zusammenarbeit im Team. Dieser Erfahrungsbericht zeigt eine Alternative unter Verwendung eines eigenen Programmierspiels auf Basis des CodinGame SDKs auf, das erfolgreich in einem Modul der Verwaltungsinformatik eingesetzt wurde.

Keywords: Programmierwettbewerb, Game-Based Learning, CodinGame, Educational Reality, Erfahrungsbericht, Präsenzlehre

1 Ausgangslage und Zielsetzung

Im Studiengang Verwaltungsinformatik der Hochschule des Bundes für öffentliche Verwaltung werden Beamtenanwärter auf Ihre späteren Aufgaben im Dienst vorbereitet. Dies erfolgt in derzeit vier parallelen Kursen je Jahrgang mit je ca. 25 Studierenden, welche verschiedene Module zu Themen der Informatik, der Verwaltungslehre und des Verwaltungsrechts absolvieren. Ein Bestandteil ist dabei auch die Vermittlung grundlegender Mechanismen der Anwendungsentwicklung sowie die Befähigung, diese umzusetzen. Dazu zählt insbesondere das Modul „Objektorientierte Programmierung“, in dem entsprechende Ansätze mit der Programmiersprache Java vermittelt werden.

Die Studierenden dieses interdisziplinären Studiengangs sind sehr heterogen in Bezug auf Vorkenntnisse der Programmierung sowie ihren persönlichen Zielen für spätere Aufgaben. Dadurch ergibt sich im genannten Modul die Herausforderung, möglichst viele Studierende zur aktiven Programmierung und tieferen praktischen Umsetzung zu motivieren. Dies wird seit mehreren Jahren mit Hilfe eines Programmierprojektes in Gruppen zum Ende des Moduls nach der Vermittlung der Grundlagen adressiert.

Diese Projekte wurden bis zum Jahr 2018 mit Hilfe der Java-basierten Programmierung von Lego Mindstorms [Le20] durchgeführt. Während damit zwar im Wesentlichen sehr gute Erfahrungen gemacht wurden und dies zur Motivation einiger Studierender beitrug, wurden jedoch auch nachteilige Aspekte beobachtet. So konnten z.B. mechanische Seiteneffekte zu Frustrationen durch unerwartetes Verhalten des Roboters führen. Hinzu

[1] Hochschule des Bundes für öffentliche Verwaltung, Fachbereich Finanzen, Verwaltungsinformatik, Gescherweg 100, 48161 Münster, {patrick.stalljohann, maik.merten}@vit-bund.de

kommt, dass die Anzahl der parallelen Kurse vorübergehend für einen Jahrgang auf sechs anwächst, um anschließend auf zwei parallele Kurse mit zwei Startterminen im Jahr zurückzufallen. Für diese Spitze sollte die Anschaffung weiterer Roboter vermieden werden, da diese auch nur einmalig genutzt werden würden.

Die Umsetzung einer Alternative sollte daher nicht von zusätzlicher Hardware abhängig sein. Außerdem soll sich jeder der Studierenden in einem Projektteam auch selbständig und ohne Abhängigkeit zur Hardware-Verfügbarkeit mit dem Projekt weiter beschäftigen können. Zu diesem Zweck sollte ein Programmierspiel zum Einsatz kommen, bei dem die Teams spielerisch und kompetitiv motiviert werden, aktiv mit Java zu programmieren.

Nach einer knappen Betrachtung der Problemstellungen von Programmierspielen wird die Programmierumgebung VITMaze vorgestellt, welche ein thematisch zur Verwaltungsinformatik passendes Programmierspiel realisiert. Es folgen genauere Beschreibungen der Spielregeln in mehreren Stufen (Level), eine Erläuterung der Projektphase sowie eine Darstellung des Finales, bei dem die Beiträge aller Teams gegeneinander angetreten sind. Abschließend erfolgen eine Bewertung der Durchführung sowie ein Ausblick auf nächste Schritte und mögliche Anpassungen sowie Erweiterungen.

2 Programmierspiele mit CodinGame

Programmierspiele sind computerbasierte Spiele, bei denen die Steuerung des Spielverlaufs nicht direkt durch ein Eingabegerät erfolgt, sondern durch die Ausführung von Programmcode, der von den Spielern eingegeben wird. Sie zählen zur Kategorie der Serious Games.

Eine von mehreren verfügbaren Plattformen ist CodinGame [Co20], auf der zahlreiche Programmierspiele sowohl im Einzelspielermodus als auch in sogenannten Arenen für mehrere Spieler bereitgestellt werden. Dabei wird Programmcode in verschiedensten Programmiersprachen (Java, C#, Python etc.) unterstützt und die Ausführung mit spielerischen Themen grafisch visualisiert. Diese Plattform kann sowohl zum privaten Vergnügen als auch zur Motivation von Studierenden eingesetzt werden [Bu16].

Bei der direkten Verwendung der Plattform durch Studierende, studienbegleitend zur Präsenzlehre, sind bereits positive Erfahrungen bekannt, wobei sich auch Defizite zeigen [HS19]. Hierzu gehört zum Beispiel, dass die Code-Qualität auf der Plattform völlig vernachlässigt werden kann, da allein das Ausführungsergebnis entscheidend ist. Außerdem ist keine Unterstützung von Entwicklungsprozessen im Team vorhanden, da ein ausschließlicher Einzelnutzerbetrieb vorhanden ist. Hierbei erfolgt die Lösung eines Programmierspiels auch grundsätzlich auf Basis einer einzelnen Code-Datei, so dass wichtige Aspekte der Softwarearchitektur, des Entwurfs und der Modularisierung nicht berücksichtigt werden können [HS19]. Mit dem Fokus auf die objektorientierte Programmierung in Gruppen sollte daher eine alternative Verwendung gefunden werden.

3 VITMaze

CodinGame unterstützt die Entwicklung neuer Programmierspiele für die Plattform aus der Community durch ein SDK [Co20a]. Der Quellcode des SDKs sowie die Vorlagen für neue Spiele stehen über GitHub zur Verfügung. Kompilierte Versionen können direkt als Pakete aus einem Maven Repository eingebunden werden. Ein neues Spiel wird durch die Implementierung entsprechender Klassen für die Spielelogik eines Spielers (Player) und der Bewertung (Referee) in einer vorgegebenen Vererbungshierarchie realisiert. Testläufe des Spiels können mit der Infrastruktur des SDKs ausgeführt werden, wobei die erzeugte Visualisierung in einer json-Datei aufgezeichnet wird, die in einer enthaltenen Weboberfläche abgespielt werden kann.

Für die Verwendung im Modul zur objektorientierten Programmierung wurde dieser Ansatz genutzt, um das Spiel VITMaze zu realisieren. Dieses Spiel wurde allerdings nicht auf CodinGame bereitgestellt, sondern den Studierenden mit einer Anpassung des SDKs als Ausführungsumgebung zur Verfügung gestellt. Anstatt eines eingeschränkte Online-Editors, sollen eine vollständige IDE verwendet werden können. Dafür wird der Spieler-Code in Form eines Java-Pakets als JAR-Datei bereitgestellt, das mehrere Dateien, Klassen und Referenzen unter einander enthalten kann.

Thematisch sollte das Spiel zur Motivation beitragen und einen gewissen Bezug zur Verwaltungsinformatik besitzen. VITMaze handelt deshalb von der Antragstellung im behördlichen Kontext und Abgabe benötigter Formulare in korrekter Reihenfolge. Spielerisch wurde dies als Verwaltungslabyrinth umgesetzt, in dem sich die Spieler zurechtfinden müssen (siehe Abb. 1). Das zugehörige Spielfeld besteht aus mehreren Feldern unterschiedlicher Art. Das Ziel des Spiels ist es, ein Java-Programm für einen Spieler/Bot zu entwickeln, der sich selbständig in unbekannten Labyrinthen bewegen kann, Formulare einsammelt und diese als erster zum zuständigen Sachbearbeiter bringt.

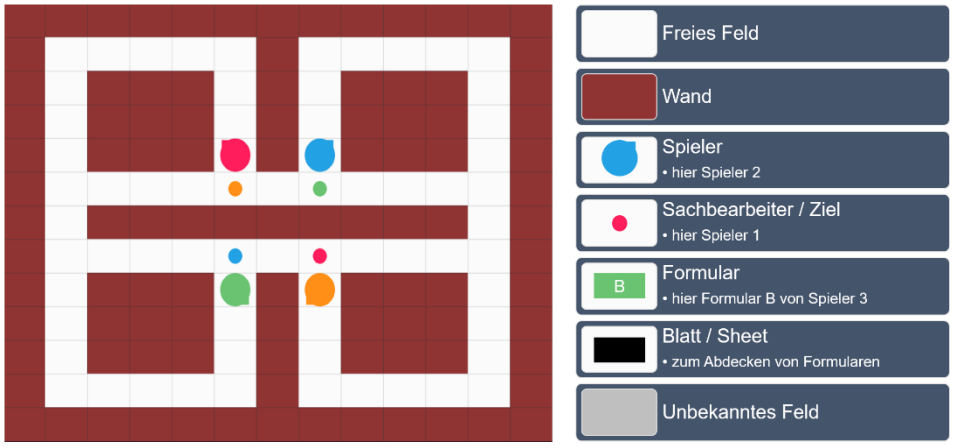

Abb. 1: Beispiel-Spielfeld (Labyrinth) und Legende zu möglichen Feldern.

4 Level – Spielregeln und Schnittstellen

Das CodinGame SDK sieht die Standard-Eingabe/Ausgabe als plattformneutrale Schnittstelle vor, auf der ein Textprotokoll zum Austausch von Kommandos und Informationen aufsetzt. Ein Bot liest zunächst die initialen Informationen über die Standardeingabe ein. Anschließend werden die Spielzüge rundenbasiert ausgeführt, so dass jeder Bot weitere Informationen der aktuellen Runde einlesen kann, um daraufhin eine entsprechende Ausgabe zu erzeugen. Diese wird wiederum von der Spielumgebung eingelesen und gemäß den Regeln ausgeführt. Anschließend werden die neuen Ergebniszustände als Informationen der nächsten Runde an die Bots weitergegeben, sofern das Spiel noch nicht beendet wurde. Eine mögliche Kapselung des Protokolls in einen objektorientierten Wrapper wurde bei der Bewertung des Systementwurfs berücksichtigt.

Das Spiel kann in fünf aufeinander aufbauenden Schwierigkeitsstufen (Level) gespielt werden, bei denen die jeweils ergänzenden Spielregeln mit Protokollerweiterungen verbunden sind. Die initialen Informationen für den Bot enthalten die Anzahl der Felder in horizontaler und in vertikaler Richtung, das Level des aktuell ausgeführten Spiels, die jeweilige ID des Bots sowie seine Startkoordinaten. Zu Beginn jeder Runde werden dazu noch das Ergebnis der Aktion aus der Vorrunde sowie der Feldstatus der aktuellen Position sowie des nördlichen, östlichen, südlichen und westlichen Nachbarfeldes bereitgestellt. Die möglichen Status sowie daraus resultierenden Aktionen, die per Ausgabe an die Spielumgebung gemeldet werden können, variieren je nach gespieltem Level.

In der einfachsten Schwierigkeitsstufe (Level 1 – Finden), besteht die einzige Aufgabe des Bots darin, durch das Labyrinth des Spielfeldes zu navigieren. Die Felder mit dem Status `WALL` (Wand) können nicht betreten werden. Die sonstigen Felder sind `FLOOR` (freier Gang) und Sachbearbeiter-Felder (`FINISH <playerId> 0`). Gültige Aktionen je Zug sind die Bewegung in eine Himmelsrichtung (`go north|east|south|west`), die Bestimmung der aktuellen Position (`position`) und das aktive Beenden beim Feld des eigenen Sachbearbeiters (`finish`). Als Ergebnis einer Aktion wird in der Folgerunde ein positiver (`OK`) oder negativer (`NOK`) Aktionsstatus an den Bot übermittelt.

In der nächsten Stufe (Level 2 – Sammeln) müssen vor Beenden der Runde alle benötigten, eigenen Formulare in richtiger Reihenfolge eingesammelt werden. Dafür wird der weitere mögliche Feldstatus `FORM <playerId> <formId>` ergänzt. Eigene Formulare können mit der zusätzlichen Aktion `take` aufgenommen werden, wenn alle Formulare mit geringerer Id auch schon aufgenommen wurden. Die Anzahl der benötigten Formulare ist zu Spielbeginn nicht bekannt, aber nun im Feldstatus des Sachbearbeiters enthalten.

Die Kollision mehrerer Bots in einem Spiel ist ab Level 3 (Unterhalten) möglich. Landen zwei Bots nach einem Zug auf dem gleichen Feld, ist die Aktion im nächsten Zug nicht erfolgreich (`NOK TALKING`) und führt zu Punktabzug. Andere Bots können aber nun auch „gesehen“ werden, indem jedem Feldstatus ein Ausrufezeichen mit Distanzwert nachgestellt ist, falls sich in dieser Richtung ein anderer Bot auf dem Gang befindet.

Aktive Störaktionen sind in den letzten beiden Stufen (Level 4 – Kicken und Level 5 – Verdecken) möglich. Dabei erlaubt die Aktion `kick` zunächst das Verschieben eines gegnerischen Formulars auf ein freies Nachbarfeld. Bei Spielbeginn mit Blättern (Sheets) ausgestattet, können diese mit der Aktion `put` auf ein Spielfeld abgelegt werden und müssen zunächst aufgehoben (`take`) oder weitergeschoben (`kick`) werden, um darunter nach Formularen zu schauen.

5 Projektphase

Die erste Projektphase mit VITMaze im Modul zur objektorientierten Programmierung wurde im Jahr 2019 mit 10 Zeiteinheiten à 90 Minuten über einen Zeitraum von drei Wochen durchgeführt. Beides erfolgte in 24 4er-Gruppen, bei denen sich je 2 Studierende frei zu Teams finden konnten, die randomisiert zu 4er-Gruppen zusammengesetzt wurden.

Zunächst wurden das Projekt, dessen Ablauf sowie die Spielregeln von Level 1 vorgestellt. Gefordert wurde außerdem die Anfertigung einer Dokumentation zu Entwurfsentscheidungen, die zusammen mit einem ausführbaren Bot, dem Quellcode und der JavaDoc-Dokumentation bis zum Ende des Projekts eingereicht werden musste. Die Bewertungskriterien zu Funktionalität und Code-Qualität auf deren Basis diese Projektarbeit zu 10% in die Modulnote einging, wurden ebenfalls bekanntgegeben. Der Umgang mit sich ständig ändernden Anforderungen wurde simuliert, indem die erweiterten Regeln und Schnittstellenbeschreibungen der Level 2 bis 5 zu vorher terminierten Zeitpunkten ergänzend bereitgestellt wurden.

Nach Bereitstellung der Ablaufumgebung und Beispiellabyrinthen mit einfachen und komplexeren Gegebenheiten, konnte die Erstellung der Bots in den Gruppen starten. Viele Gruppen haben dazu ein GitHub-Repository angelegt, um gemeinsam und versioniert am Code zu arbeiten. In den Präsenzveranstaltungen wurden die Studierenden bei der Entwicklung betreut und mit Anregungen sowie der Beantwortung von Fragen unterstützt. Neben diesen Präsenzzeiten konnten die Studierenden weiter frei an Ihrer Lösung weiterentwickeln.

6 Das Turnier

Zum Abschluss des Projektes traten die 24 Bots gegeneinander an. In jeder Stufe des Turniermodus (Battle) wurden 3 verschiedene Begegnungen (Matches) in unterschiedlichen Labyrinthen gespielt. Zunächst wurden je 4 Bots zufällig zu 6 Start Battles zugeteilt, in den bekannte Labyrinthe in Level 2 gespielt wurden. Die beiden besten Bots je Start Battle erreichten die drei Medium Battles, in denen unbekannte Labyrinthe in Level 3 anstanden. Die wiederum zwei besten Bots gelangten in die drei Rough Battles in Level 4, bei denen je zwei Spielerbots von zwei Gegnerbots gestört wurden. Die drei Gewinner der Rough Battles gelangten in das gemeinsame Finale, das mit Regeln nach

Level 5 gespielt und durch einen Gegner-Bot gestört wurde. Der Gewinner des Finales wurde zum Siegerbot gekürt und die zugehörige Gruppe erhielt einen Wanderpokal.

Die Durchführung erfolgte nach Abgabe der Bots, so dass die verschiedenen Begegnungen ausgelost und durchgeführt wurden. Die vorab berechneten Aufzeichnungen der Turnier-Matches wurden in einer Audimax-Großveranstaltung den Teilnehmenden, Lehrenden und Studierenden aller Semester vorgestellt. Hierbei wurde der Turnierverlauf mit einer eigens erstellten Webanwendung visualisiert, so dass "mitgefiebert" werden konnte.

7 Feedback, Bewertung und Ausblick

Der sehr positive Eindruck des Autors wurde durch das Feedback der Studierenden im direkten Gespräch, Feedbackrunden und Evaluationsbögen des gesamten Moduls durchgängig bestätigt. Es war zu beobachten, dass die Studierenden sehr viel Zeit mit der aktiven Programmierung des Projektes verbracht haben, auch außerhalb der Präsenzzeiten. Insbesondere wurden aber auch Entwürfe entwickelt sowie geplant und modular programmiert, da von Anfang an klar war, dass Code-Qualität und Struktur in die Bewertung eingehen, das Abschneiden beim Turnier jedoch nicht. Dennoch schien das anstehende Turnier die Motivation vieler Gruppen deutlich zu stärken. Dabei konnten die Studierenden den Umgang mit ihrer Entwicklungsumgebung in der Gruppe (Eclipse + Git) weiter trainieren und auch objektorientierte Konzepte in mehreren Dateien für ihre Abgabe nutzen.

Die größte Anregung der Studierenden war, das Debugging in der laufenden Spielumgebung zu ermöglichen. Aus Sicht des Autors wäre ein Redesign hin zu einer Serverumgebung mit einem service-basierten API wünschenswert, in der der Entwicklungsfortschritt direkt verfolgt werden kann, um eine bessere Transparenz des Lernfortschritts zu ermöglichen.

Literaturverzeichnis

[Bu16] Butt, P.: Students' perceptions of game-based learning using CodinGame. In (Morris, L. und Tsolakidis, C., Hrsg.): International Conference on ICT in Education, Rhodes, Greece, S. 151–158, 2016.

[Co20] CodinGame, Coding Games and Programming Challenges to Code Better, www.codingame.com, Stand: 18.03.2020.

[Co20a] CodinGame SDK Documentation, https://www.codingame.com/playgrounds/25775, Stand: 19.03.2020

[HS19] Hinrichs, T.; Schmolitzky, A.: Einsatz einer Online-Programmierplattform in der Präsenzlehre – ein Erfahrungsbericht. In (Strickroth, S. et al., Hrsg.): 4. Workshop "Automatische Bewertung von Programmieraufgaben" (ABP2019), S. 59–62, 2019.

[Le20] Lego Mindstorms Education EV3, https://education.lego.com/de-de/product/mindstorms-ev3, Stand: 18.03.2020