Technische Hochschule Deggendorf

Fakultät Angewandte Informatik
(Bachelor-Studiengang Angewandte Informatik)

# JAVA Programmierung FWP - 2D Plattformer

Projektarbeit
für das Fach
Javaprogrammierung AI-B6-FWP
SS2022

vorgelegt von:

Robin Prillwitz (00805291), 84326 Falkenberg

am: 1. Juli 2022

Erstprüfer:

_____

# Inhaltsverzeichnis



# Abkürzungsverzeichnis

| | |
|---|---|
| **CSV** | Comma Seperated Values. |
| **HTML** | HyperText Markup Language. |
| **JAR** | Java Archive. |
| **JDK** | Java Development Kit. |
| **JRE** | Java Runtime Enviornment. |
| **UML** | Unified Modelling Language. |

# Glossar

| | |
|---|---|
| **Javadoc** | Java's standard Dokumentationsmethode. |
| **JavaFX** | Java Graphikbibliothek. |
| **Maven** | Build System. |
| **MySQL** | Relationale Datenbank. |

# 1 Allgemeine Übersicht

*Domino's Pizza® dominiert die Dystopie Düsburgs. Kann Pizza Hut® den Ruf der runden Fressalien retten? Begib dich zur Auslieferung an die Front und beweise deine italienischen Wurzeln!*

Das Spiel *Pizza Hut 2077* ist ein Sidescolling Jump'n'Run. Das Spiel besteht aus einer simplen 2-Dimensionalen Welt. Diese ist mit zufällig generierten Plattformen gefüllt. Das konkrete Ziel des Spiels besteht lediglich in dem Highscore. Während des Ablaufes muss der Spieler Gegnern ausweichen und sich von Platform zu Platform hangeln. Der Spieler kann diese Gegner durch das Werfen von Pizzen eliminieren. Die Feuerrate der Pizzen ist jedoch limitiert. Das Spiel ist zu Ende, wenn der Spieler von einem Gegner berührt wird oder aus der Welt fällt (bzw. nicht auf einer Platform landet). Alle dem Genre üblichen Funktionen sind wie zu erwarten realisiert.

Am Ende des Spiels kann der Spieler seinen Highscore mitsamt einem Namen in einer Datenbank speichern. Die *MySQL* Datenbank ist online gehosted, darum wird eine funktionierende Internetverbindung benötigt. In dem Credits-Bildschirm kann, mit dem notwendigen Administrator Passwort, die Highscore Tabelle der Datenbank zurückgesetzt werden. Das einfügen von neuen Highscores zum Spielende verläuft ohne Passwort. Die Dateneingabe hat jedoch nur Rechte zum Erstellen neuer Daten und ist auch gegen `SQL`-Injections gehärtet.

Die Hintergrundmusik des Spiels reagiert auf den jeweiligen Zustand. Im Hauptmenü wird eine ruhigere Version des Arpeggio Leitmotivs gespielt. Zum Spielstart geht diese mit über eine Brücke in den Hauptteil über. Der Hauptteil wird während des Spielverlaufs gelooped. Wenn der Spieler stirbt, wird der Hauptteil durch das Outro unterbrochen. Der Soundtrack stammt aus eigener Komposition. Außerdem haben die Aktionen des Spielers zugehörige Soundeffekte, wie beispielsweise beim Springen oder bei dem Werfen von Pizzen, die passend abgespielt werden.

Die Graphik des Spiels beruhen auf *JavaFX*. Die meisten graphischen Elemente und Animationen sind durch individuelle Sprites implementiert. Einfache Animationen wurde durch das Abspielen verschiedener Sprites nacheinander realisiert. Alle Sprites stammen aus eigener Produktion. Interaktive Elemente wie Text und Eingabefelder nutzen jedoch *JavaFX* Widgets.

Das Projekt wird mithilfe von *Maven* kompiliert. Zur Ausführung ist eine funktionierende Instanz davon nicht notwendig, lediglich eine *Java Runtime Enviornment (JRE)* von Version 18 oder später. Da das Projekt in eine eigenständige *Java Archive (JAR)* mitsamt aller Ressourcen (Sprites und Sounds) und aller notwendigen Abhängigkeiten (*JavaFX* und Freunde) kompiliert werden kann. Außerdem kann durch *Maven* eine vollwertige und interaktive Dokumentation mit *Javadoc* erstellt werden. Die gesamte Codebase ist auch in der *Javadoc* Konvention kommentiert.

Anleitungen zur Kompilation, Steuerung und Tasten und weiteres können in der `README.md` gefunden werden. Attribution für benutzte Ressourcen ist in der `ATTRIBUTION.md` gelistet.

## 1.1 *UML*-Diagramm

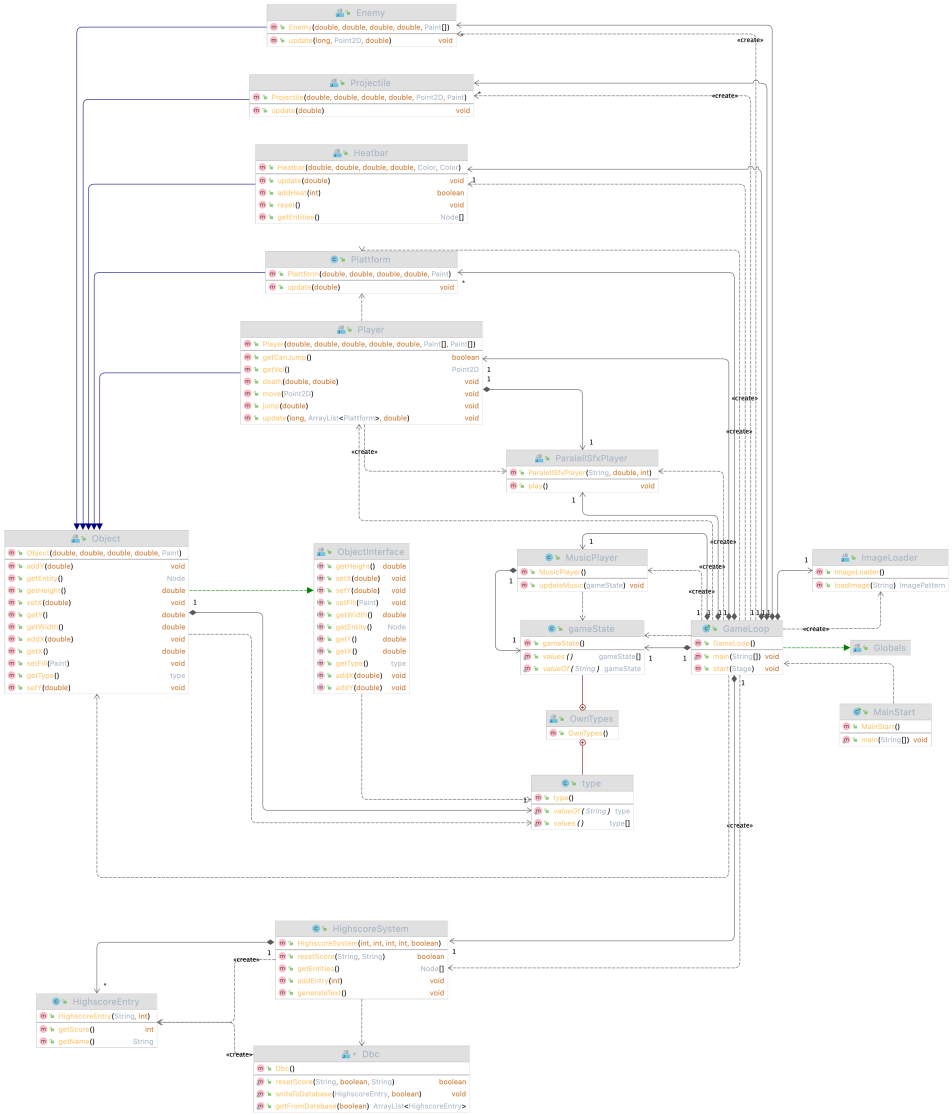


**Abbildung 1:** *Unified Modelling Language (UML)*-Diagramm des Projekts. Das Bild befindet sich auch im Haupt-Verzeichniss des Projekts.

Das ist Abbildung 1 dargestellte *UML* Diagramm gibt den gesamten Inhalt des implementierten Packets an. Äußere Abhängigkeiten werden nicht aufgeführt. Der Großteil spielt sich in und um der Klasse `GameLoop` ab. Die relevanten Teile der Core-Gameplay-Schleife und weiteres werden darin erstellt und aktualisiert. Alle im Spiel auftretenden Elemente erben von der Klasse `Objekt`, welche das `ObjectInterface` implementiert. Musik wird durch die `MusicPlayer` Klasse gehandhabt und Sound durch den `ParrallellSFXPlayer`. Bilder und Graphiken werden durch den `ImageLoader` geladen aber von den Verbrauchern selbst verwaltet. Das Highscore System mitsamt der Datenbank Verbindung wird in `HighscoreSystem`, `HighscoreEntry` und `DBC` (DataBase Connection) implementiert. Die Klasse `MainStart` ist eine helferklasse um die *JavaFX* Applikation zu initialisieren.

# 2 Eigene Beiträge

Viele Teile des Programms, vor allem Teile der größeren Klassen, wie von `GameLoop` sind durch Kollaboration entstanden. Kein Teil wurde ausschließlich in Isolation entwickelt. Daher überdecken sich einige Teile möglicherweise, die genaue Einteilung wurde daher nach unserem besten Gewissen festgelegt.

## 2.1 Highscore & Datenbankverbindung

Das Highscore System und die Datenbankverbindung ermöglichen es einen Highscore sowohl zu speichern, als auch die $n$ größten Highscores anzuzeigen. Dabei ist $n$ in `Globals` frei einstellbar[1]. Die Anzeige der Highscores erscheint zum Ende eines Spiels, wenn entweder weniger als $n$ Highscores existieren oder der erreichte Score größer als der kleinste ist. Erfüllt der erreichte Score diese Kriterien, so wird der Spieler aufgefordert seinen frei wählbaren Namen anzugeben[2]. Die Angabe des Namens ist jedoch optional, das Spiels kann auch ohne Eintragung neu gestartet werden. Die Highscore Daten können entweder in eine Online-Datenbank oder offline in eine *Comma Seperated Values (CSV)* Datei gespeichert werden. Standardmäßig ist die Offline *CSV* Version eingestellt. Das kann wieder in `Globals` verändert werden. Die Implementation dieser gesamten Funktionalität konzentriert sich auf die Klassen `Dbc` (kurz für *DataBase Connection*), `HighscoreSystem` und `HighscoreEntry`. Außerdem hat das System Einzugsbereiche im `GameLoop`. Zu einem wird die Anzeige daraus gestartet und erhält den aktuellen Score. Zum anderen kann die Datenbank, egal ob online oder offline, durch eine Passworteingabe im *Credits* Bildschirm zurückgesetzt bzw. gelöscht werden. Für die Online-Datenbank wird das Administrator Passwort der Datenbank benötigt.Für die Offline-Datenbank ist es das `DefaultPassword` in `Globals`.

Die Online Datenbank ist eine *MySQL* Datenbank, welche mit `preparedStatements` arbeitet. Die Offline Datenbank benutzt *Apache*'s `commons-csv` Bibliothek um die Daten in eine *CSV* Datei zu schreiben. Die Implementation befindet sich in `Dbc`. Die Klasse `Dbc` ist final und hat keinen internen Zustand. Somit werden zwar mehr Ressourcenaufrufe benötigt, jedoch wird sichergestellt, dass die angezeigten Daten stets so aktuell wie möglich sein können.

Die Klasse `HighscoreEntry` stellt einen einzelnen Highscore, ein Datenpaar aus Name und Score dar. Diese werden im `HighscoreSystem` als Arraylist gespeichert. Dort werden diese Daten und auch das Namenseingabefeld und die Überschrift graphisch angezeigt. Die Texteingabe und Validierung in das Namensfeld wird auch darin gehandhabt. Die `Dbc` wird folgend davon aufgerunfen um entweder einen neuen Highscore zu schreiben, alle geschriebenen auszulesen oder um alle zu löschen.

[1] Die Anzahl ist standardmäßig $n = 10$.
[2] Die Länge des Namens ist auch in `Globals` standardmäßig auf 10 Zeichen limitiert.

## 2.2 Build-System

Das Build-System des Projekts stellt das System zur Kompilation, Ausführung, Verpackung, Abhängigkeits management und der *Javadoc* Dokumentation dar. Das Projekt benutzt dafür *Maven*. Die Konfigurationsdatei dafür ist `pom.xml`. Die Kommandos (auf der Kommandozeile) aus Tabelle 1 werden dadurch zur Verfügung gestellt.

*Maven* kümmert sich außerdem um die externen Abhängigkeiten, wie *JavaFX*, `commons-csv` und `mysql-connector-java`. Dabei werden die angegebenen Versionen bei bedarf automatisch aus dem *Maven* Repository heruntergeladen.

Die komilierten Dateien und die *JAR* Datei sind im `./target` Underordner anzufinden. Es werden mehrere *JARs* erstellt. Die relevante ist unter `./target/SidescrollerGame-jar-with-dependencies.jar`. Aufgrund der *JavaFX* Abhängigkeiten ist diese leider nicht Platformagnostisch. Ergo kann diese nur auf dem Betriebssystem / der Prozessorarchitekur auf der sie erstellt wurde ausgeführt werden [3]. Die *JAR* kann jedoch ohne *Java Development Kit (JDK)*, nur mit *JRE*, ausgeführt werden, da alle Abhängigkeiten und Ressourcen inkludiert werden. Das hebt jedoch die Dateigröße auf rund 27 Megabyte an.

Die *Javadocs* ist in `./javadoc` anzufinden. Diese beziehen sich dabei auf die *Javadoc* Kommentare im Code. Außerdem wird zu jeder Klasse in der interaktiven *HyperText Markup Language (HTML)* Dokumentation ein kleines *UML*-Diagramm generiert. Die passiert durch das `umldoclet` Plugin von `talsmasoftware`. Die Dokumentation kann mit `./javadoc/index.html` durch jeden Browser geöffnet werden.

| Kommando | Bedeutung |
|---|---|
| `mvn clean` | Löscht alle Build-Atrefakte |
| `mvn compile` | Kompiliert das Projekt |
| `mvn package` | Kompiliert und Erstellt eine *JAR* Datei |
| `mvn exec:java` | Kompiliert und führt das Projekt aus |
| `mvn javadoc:javadoc` | Erstellt die interaktive *HTML* Dokumentation |

**Tabelle 1:** *Maven* Kommandos

## 2.3 Graphik & Animationen

Alle Bilder und Animationen werden durch Sprites implementiert. Diese sind im `sprites` Unterordner anzutreffen. Sie wurden durch *Adobe Photoshop*, *After Effects* und *Media Encoder* erstellt. Bilder könenn durch den `ImageLoader` geladen werden. Diese greift auf Dateien zu, daher stand die Implementation sehr in Verbindung mit dem Build-System. Es musste sichergestellt werden, dass die zu ladenenden Bilder auch in der *JAR* zugänglich sind.

Der `ImageLoader` läd Sprites so, dass diese durch die implizite Skalierung bei der Anwendung der Sprites auf ein Graphikobjekt ohne Anti-Aliasing skaliert werden. Dadurch ersteht der Ef-

[3] Dies wurde getestet zwischen einem x86 Windows 10 Computer und einem Apple M1 MacBook Pro.

fekt, dass alle Sprites, egal wie groß, eine annehmbare Größe annehmen und dabei auch piexeliert werden. Das hilft sofern, dass eigene Piexl-Art tatsächlich auf den Dimensionen der Piexl gezeichnet werden kann und zum Export nicht skaliert werden muss. Beispielsweise können die Spieler Sprites im Format von $16 \times 16$ Pixel gezeichnet und gespiechert werden, die Hintergründe können aber wesentlich größer sein. Im Spiel nehmen beide jedoch eine vergleichbare Sklaierung an.

Animationen zwischen mehrern Sprites können im Spieler, den Projektielen, den Gegener und im Hauptmenü gefunden werden. Die Animationen bestehen aus einem einfachen Timer, welcher nach definierter Zeit die Sprite wechselt. Der Spieler verfügt über zwei verschiedene Animationszustände: Ein *Idle* und ein *Running* Zustand. Je nach der momentanen geschwindigkeit des Spielers werden die stationäre *Idle* Animation oder die bewegende *Running* Animation benutzt.

## 2.4 Musik & Sounddesign

Die Musik wurde in *Ableton Live* komponiert und in *Adobe Audition* gemischt und gemastert. Die Musik wurde für das Spiel in vier Teilen arrangiert: Zuerst ein wiederholbarer Intro-Abschnitt, gefolgt von eienr Brücke, die den Übergang zum wiederholbarern Hauptteil bildet und schließlich einen Schlussteil, welcher den Hauptteil zu jedem Zeitpunkt unterbrechen kann. Die einzelnen Sektionen wurden dannach einzelnen exportiert.

Die Soundeffekte wurden auch in *Ableton Live* erstellt. Es gibt Effekte für springen, projektiele werfen, gegener zerstören und punkte erlangen. Die Lautstärke jedes Soundeffekts wurde im Code so angepasst, dass diese sich in die Klanglandschaft integrieren und nicht dominieren. Alle Sounds und Musik ist im `./sound` Ordner gespeichert.

# Eidesstattliche Versicherung

Ich, **Robin Prillwitz, 00805291**
(Vorname, Name, Matr.-Nr.)

versichere an Eides Statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder dem Sinne nach aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe.

Ich versichere an Eides Statt, dass ich die vorgenannten Angaben nach bestem Wissen und Gewissen gemacht habe und dass die Angaben der Wahrheit entsprechen und ich nichts verschwiegen habe.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 163 Abs.1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

**84326 Falkenberg, 1. Juli 2022**
Ort, Datum


Unterschrift


## Kolophon

Dieses Dokument ist ein LaTeX 2ε (2022-06-01) Dokument der KOMA-Script Klasse. Alle eigenen Zeichnungen sind mit Ti*k*Z gesetzt. Kompiliert wurde es mit XeLaTeX und biber *2.17* mithilfe von TeX Live auf macOs 12.4 am 1. Juli 2022.