TECHNISCHE HOCHSCHULE DEGGENDORF

FAKULTÄT ANGEWANDTE INFORMATIK

Prüfungsstudienarbeit zum Thema

JAVA Programmierung FWP – 2D Plattformer

Abbildung 1: Titelbildschirm

vorgelegt von

Simon Obermeier – Matrikelnummer: 00800498

Gruppenmitglieder

Robin Prillwitz – Matrikelnummer: 00805291

Anton Kraus – Matrikelnummer: 00804697

Eingereicht: 30.06.2022

Betreuer: Nicki Bodenschatz

Inhalt

Abkürzungsverzeichnis

CSV	Comma Seperated Values
HTML	Hyper Text Markup Language
JAR	Java Archive
PDF	Portable Document File
SFX	Sound Effekts
UML	Unified Modeling Language

Erklärung

Hiermit versichern wir, Anton Kraus und Simon Obermeier, diese Dokumentation ohne die Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht worden.

Ort, Datum, Simon Obermeier

Allgemeines zum Programm

In unserer Prüfungsstudienarbeit im FWP Java-Programmierung haben wir uns dafür entschieden einen kleinen 2D-Plattformer mit der Bild-Engine „JavaFX“ umzusetzen. In dem Spiel geht es darum, als Pizzabote über zufällig generierte Plattformen zu springen und dabei ankommenden Pizza-Konkurrenten auszuweichen und diese auch zu eliminieren. Das Spiel an sich hat kein richtiges Ende und hat als Ziel einen Möglichst hohen Punktestand zu erreichen, um sich dann in die globale Datenbank eintragen zu lassen und unter den Top 10 Punkteständen zu erscheinen. Die Menüführung wurde dabei über Tastaturanschläge realisiert. Im Hauptmenü kann man mit „Q“ zum Spielgeschehen wechseln, oder mit „E“ die Autoren der Software nachschauen, beziehungsweise die Datenbank mithilfe des Admin Passwortes zurücksetzen. Im Spiel kann man sich dann mit „A“ und „D“ nach links und rechts bewegen. Drückt man die Leertaste springt man. Dies ist allerdings nur einmal pro Plattformberührung möglich. Mit der Maus kann man dann zielen und durch die Linke Maustaste ein Projektil auf die Gegner abfeuern. Diese Projektile sind an eine Hitzeleiste am oberen Rand des Bildschirms gebunden. Diese Leiste muss erst wieder abkühlen um neue Projektile zu schießen. Man kann in allen Bereichen mit „ESC“ eine Ebene zurückspringen oder im Hauptmenü das Spiel beenden. Die Kurzfassung dazu kann im dafür angelegten GitHub Repository angesehen werden.

Es wurde auch für die musikalische Begleitung im Projekt gesorgt. Dafür wurde ein dynamischer SFX-Player geschrieben, welcher je nach dem aktuellen Game-State die dazugehörigen Sound-Dateien abspielt und einreiht. Auch wurde dafür gesorgt, dass die Sound-Dateien nur einmal geladen werden und dann einfach dynamisch eingereiht werden. Dies reduziert den Ressourcenverbrauch des Spiels um einiges. Die Soundtracks wurden dabei mit den Programmen „Live“ von Ableton und „Audition“ von Adobe erstellt. Diese basieren aufeinander und bilden verschiedene Segmente der Musik, welche zum Beispiel beim Spielen noch mehrere Elemente zur Titelmusik hinzufügen.

Die Highscores des Spiels werden in einer Online-Datenbank gespeichert und zur Laufzeit direkt geladen. Im Spiel werden dabei immer die Top 10 Spieler angezeigt, welche in der Datenbank liegen. Im Spiel kann diese Datenbank mit dem Admin Passwort zurückgesetzt werden. Legt man im „Globals“-Interface den Paramter „onlineMode“ auf „false“, dann werden die Highscores in einer lokalen „CSV“ Datei abgespeichert. Auch kann dort das Passwort gesetzt werden mit dem die Highscores dann offline zurückgesetzt werden können (default: 123456).

Maven wird als Build-System für das ganze Projekt benutzt. Wir nutzen es um die lauffähige Jar-Datei zu packen, die Javadoc zu generieren und auch um das Spiel beim Entwickeln zu kompilieren und zu starten. Die Einstellungen sind dabei in der „pom.xml“ definiert. Dies wurde in Visual Studio Code integriert. Für die Nutzung von Maven kann die „README“ im Projekt referenziert werden. Dort steht auch nochmal die Hintergrundgeschichte zum Szenario, die Tastenbelegungen, Benutzung des Build-Systems und auch aufrufen der Javadoc.

Zusammenhänge der Klassen und Packages

Unsere Hauptklasse des Programms ist „gameLoop“, welche allerdings nicht die Main-Methode der Software enthält. Diese ist in der Klasse „mainStart“ definiert. GameLoop erbt dabei zum einen vom Interface „Globals“ und zum anderen von der JavaFX Superklasse „Applikation“. In dem Interface sind alle globalen Variablen für Spieleinstellungen definiert, um diese schnell und übersichtlich ändern zu Können. Die Superklasse Applikation wird benötigt um im Projekt ein Fenster des Frameworks zu erzeugen. Diese Zusammenhänge können auf der nächsten Seite, auf Abbildung 2, genauer betrachtet werden. Alle Zeichenbaren Objekte des Spiels, erben dabei von der Superklasse „Object“, welche wiederum das Interface „gameObject“ nutzt. Dabei wird sichergestellt, dass alle zeichenbaren Objekte des Spieles die gleich Grundfunktionalität besitzen und Fehler vermeiden. Auch lassen sich dann alle Objekte in einem Array speichern, falls man dies möchte. Wir haben uns allerdings dagegen entschieden, um Prüfungen auslassen zu können, um welches Objekt es sich letztendlich genau handelt. Die Menüführung wurde über eine interne State-Machine realisiert. Diese wechselt, je nach Input des Users oder je nach Auslösern im Spiel den State des Programms und lädt, beziehungsweise entlädt bestimmt Programmteile. Unsere Sates wurden dabei alle in einzelnen JavaFX Panes realisiert. Diese erhalten zum Programmstart alle darin enthaltenen Elemente und co-existieren zusammen mit dem GameRoot, welcher das Fenster an sich darstellt. Findet nun ein Szenenwechsel statt werden die nicht mehr benötigten Panes aus dem GameRoot entladen und die neuen dazu geladen.

Das Programm wurde auch vollständig mit Javadoc kommentiert und eine Klickbare HTML Dokumentation erstellt. Um diese Aufzurufen muss die „index.html“ unter „./javadoc/apicdocs“, mit dem bevorzugten Browser geöffnet werden. Diese Dokumentation ist völlig Interaktiv und liefert auch zu jedem Package die Abhängigkeiten der Klassen zueinander.

Erfüllung der Anforderungen

Die gestellten Anforderungen an die Software waren Vererbung, Interface und ach Javadoc Dokumentierung. Vererbung wurde zum einen für die Nutzung von JavaFX verwendet, als auch beim Abstrahieren der zu zeichnenden Spielobjekte. Ein Interface fand beim Anlegen von zentralen globalen Variablen seinen Platz. Die Kommentierung mit Javadoc wurde für alle Funktionen, Konstruktoren und auch die wichtigsten Variablen, sowohl im Private- als auch im Protected – und Public-Scope durchgeführt. Daraus wurde dann auch eine interaktive HTML Dokumentation generiert (vgl. Ende vorheriger Abschnitt).

Besondere Herausforderungen an das Programm waren dann noch zusätzlich das Verwenden einer GUI, das Erstellen einer Lauffähigen jar-Datei und das Nutzen eine Relationalen Datenbank. Des GUI wurde in diesem Projekt mit dem JavaFX Framework durchgehend realisiert. Auch die Tonkulisse und Bilder im Hintergrund wurden damit umgesetzt. Auch lässt sich aus unserem Programm zu jeder Zeit eine jar-Datei generieren. Dies geschieht über das Framework „Maven“. Hat man Maven installiert, kann man mit „mvn package“ eine Jar-Datei generieren lassen. Die Datenbank kommt zum Einsatz, um die Highscores der Spieler mit

Namen und Punktzahl global im Internet abzuspeichern. Damit können auch andere Spieler diese Highscores sehen und versuchen diese zum Überbieten.

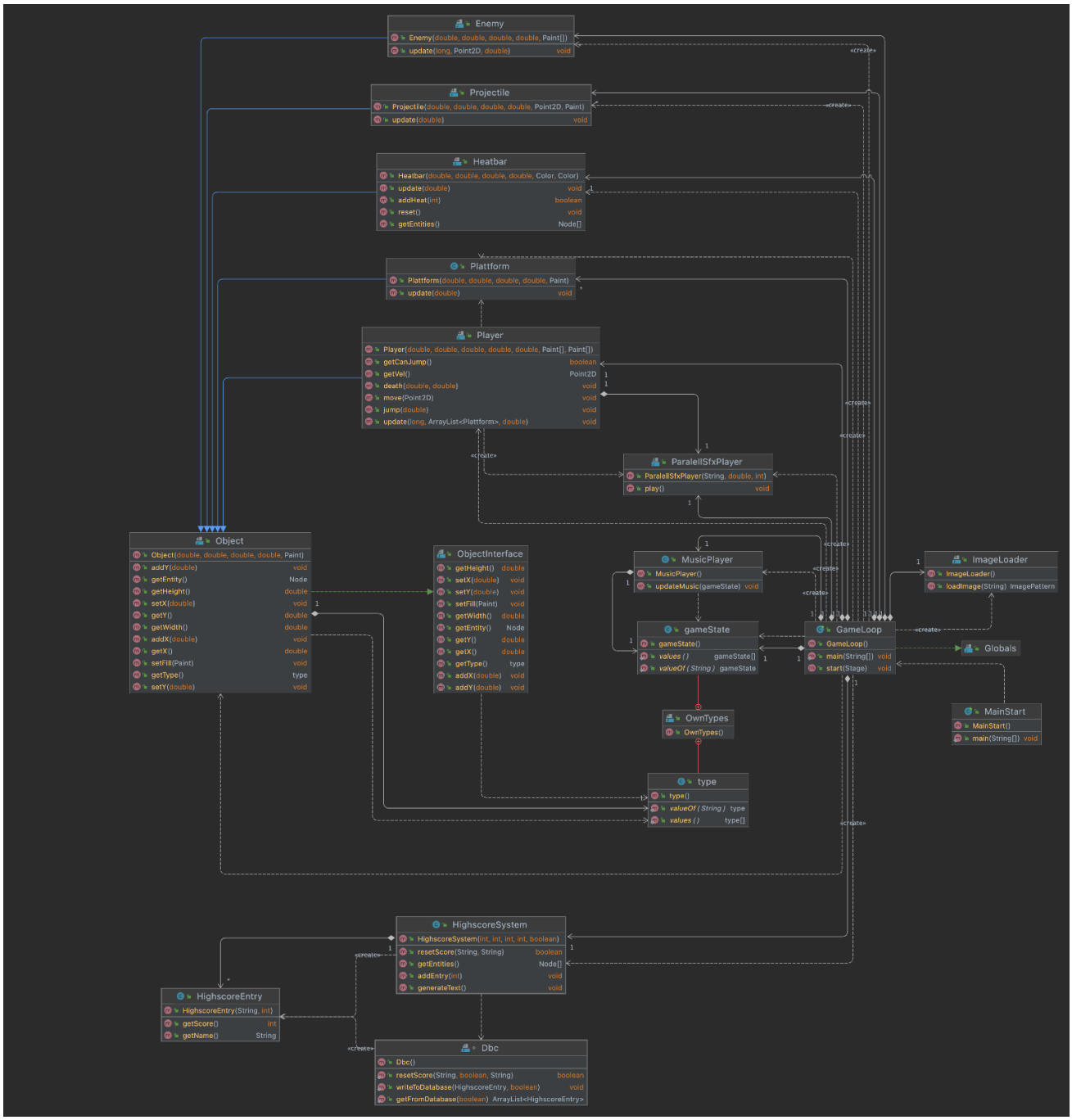

Abbildung 2: UML Klassendiagramm

Diese Abbildung kann auch im Repository, oder dem abgegeben Projekt-Ordner unter dem Namen „uml.png“ im Root Verzeichnis gefunden werden. Die Auflösung nach dem Einfügen in Word oder PDF wird leider sehr herunterskaliert und kann daher schwer leserlich sein.

Eigene Leistung am Programm

Zur eigenen Leistung am Projekt gehören alle im Spiel gezeichneten Objekte, dessen Erzeugung und mögliche Kollisionen zwischen den Objekten. Auch das Management der sogenannten „JavaFX-Panes“, um die Objekte dann auch gesammelt im richtigen Game-State anzeigen zu lassen. Bei den Javadoc Kommentaren kann man den Teil nicht auf eine Person auslagern, da jeder seinen eigenen Code immer wieder kommentiert hat.

Alle zeichenbaren Spielobjekte erben von einer Superklasse namens „Objekt“, welches wiederum ein Interface implementiert. Dieses Interface stellt alle Funktionen und Variablen dar, welche ein zeichenbares Objekt haben sollte. In der Superklasse werden dann primitive Getter und Setter Methoden implementiert und diese dann in den jeweiligen Unterklassen noch weiter spezifiziert (falls nötig). Dazu gehört auch die Prüfung für Kollisionen. Der Spieler muss mit den Gegner und die Gegner müssen mit den Projektilen kollidieren können. Auch müssen die Objekte, welche nicht mehr gebraucht werden, also außerhalb des Bildschirms sind, entfernt werden um die Performance des Spiels zu verbessern.

Die Plattformen und Gegner des Spiels werden völlig zufällig und endlos generiert. Die Plattformen haben einen festen x und y Abstandsbereich in dem sie relativ zur alten Plattform am rechten Bildschirmrand erscheinen können. Auch richtet sich dieser Wert immer am unteren Drittel des Bildschirms, da sonst die Plattformen irgendwann aus den Rändern wachsen könnten. Die Gegner hingegen können rund um den Spieler erscheinen (außerhalb des Bildschirms) mit einen festen Abstandsintervall. Die Gegner bewegen sich dann in einer direkten Linie zum Spieler hin und führen zum „sterben“ des Spielers bei Berührung.

Die Projektile des Spiels werden per Mausklick in Richtung des Mauszeigers geschossen. Dabei wird ein Vektor zwischen dem Spieler und des Mausposition als Bewegungsvektor benutzt. Die Frequenz der Projektile ist dabei durch eine „Hitzeleiste“ limitiert. Diese sammelt bei jedem Schuss Hitze an und entlädt sich immer langsam. Diese Hitzeleiste ist auch eine weiteres Spielobjekt, welches von der Superklasse Objekt erbt.

Alles Spielobjekte werden, nach State des Spiels thematisch sortiert, einem dafür zugeordneten „Pane“ hinzugefügt. Dies geschieht alles beim initialisieren des Spiels, damit nachher nichts mehr nachgeladen werden muss. Wechselt nun der State, kann diese gesamte Sammlung einfach geladen werden und man muss dies nicht immer einzeln machen.

Um Abläufe zu vereinfachen wurde auch ein zentrales Interface „Globals“ definiert. Dieses wird durch die Hauptklasse „GameLoop“ implementiert. In diesem Interface sind Variablen definiert, wie zum Beispiel die Größe des Spielers, die Sprungkraft und so weiter. All dies sind Werter, welche zum Fine-Tuning des Spiels benötigt wurden und hier zentral geändert werden können.