Konzept „Don‘t touch the wall“

Projekt im Rahmen der Lehrveranstaltung D2 Objektorientierte Programmierung

Autoren: Karin Fehringer, Ilkkan Alphan, Michael Tausch, Mario Ammann

Version: 1.3

Letzte Änderung: 19.12.2022

Inhalt

Einleitung:

Im Rahmen der Lehrveranstaltung ‚Objektorientierte Programmierung' hat sich die Gruppe entschieden ein Spiel zu entwickeln, in dem eine Maus einen vordefinierten Pfad folgen muss, ohne die Wand zu berühren. Je höher das erreichte Level ist, umso schwieriger ist es dem Pfad zu folgen und die Zeit darf dabei nicht überschritten werden. Je schneller das Ziel erreicht wird, umso mehr Punkte können generiert werden.

Funktionsbeschreibung:

Übersichtsseite

- Texteingabe für Spieler*in-Name
- Highscore der besten Spielzeiten, sortiert nach erreichter Punktezahl
- Button zum Start des Spiels

Spiel

Spielaufbau

- Mauszeiger
  Die/der Spieler*in kann den Cursor mit der Maus über den Bildschirm bewegen.

- Levelbild mit Pfad
  Jedes Level wird aus einem Bild bestehen, dass mit einer Hintergrundfarbe den Pfad beschreibt, dem die Maus folgen soll.
  Es gibt einen Start- und Endbereich in jeweils vordefinierten Farben (grün und rot), die im Levelbild eingezeichnet sind.

- Counter
  Bei Spielstart beginnt die Zeit nach unten zu zählen und wird, so wie der User*in-Name, im Fenster angezeigt.

- Hindernisse
  In den Levels gibt es verschiedene Hindernisse (Animated-Gifs), die sich bewegen und nicht berührt werden dürfen.

- Soundeffekte
  Bei Spielstart, bei Zielerreichung und bei Game-over ertönt ein Sound.

Spielablauf

- Level wird aufgebaut, Timer steht bei der maximal zulässigen Zeit des jeweiligen Levels, alle Hindernisse sind im Level platziert.

- Spieler*in bewegt die Maus auf den Startbereich, daraufhin beginnt die Zeit nach unten zu zählen und die Hindernisse beginnen sich zu bewegen.

- Die/der Spieler*in versucht mit dem Mauszeiger so schnell als möglich das Ziel zu erreichen, ohne dabei ein Hindernis oder eine Wand zu berühren.

- Wenn ein Hindernis oder eine Wand berührt wird oder die Zeit abläuft, ist das Spiel zu Ende. Das Fenster schließt sich, der Highscore wird aktualisiert und angezeigt. Die/Der Spieler*in hat die Möglichkeit von vorne zu beginnen.

- Erreicht die/der Spieler*in das Ziel öffnet sich ein Pop-up mit Erfolgsmeldung ‚You did it!‘, mit erreichter Punktezahl und „Weiter“-Button.

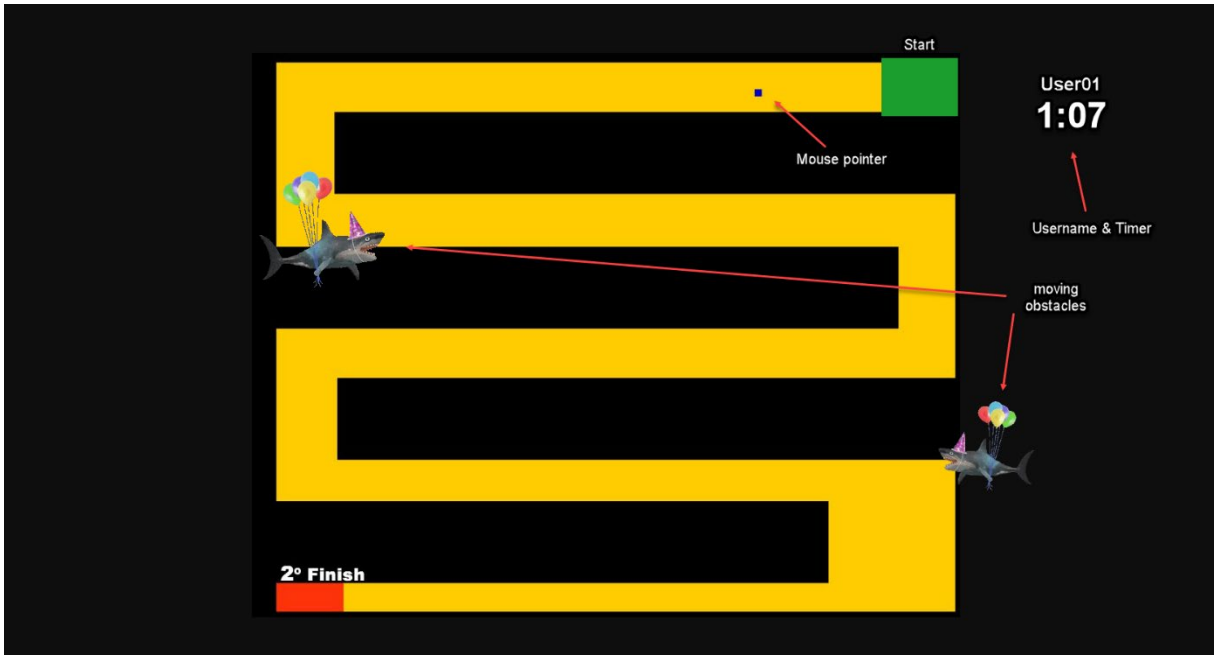
- Nach Klick auf „Weiter“-Button startet das nächste Level und das Spiel beginnt wieder.

- Bei Erreichen des letzten Levels (derzeit insgesamt 4) öffnet sich ein Gratulations-Popup, zB. Feuerwerk-Animation, mit Erfolgstext, mit Punktezahl und „Beenden“-Button.

Spielelogik

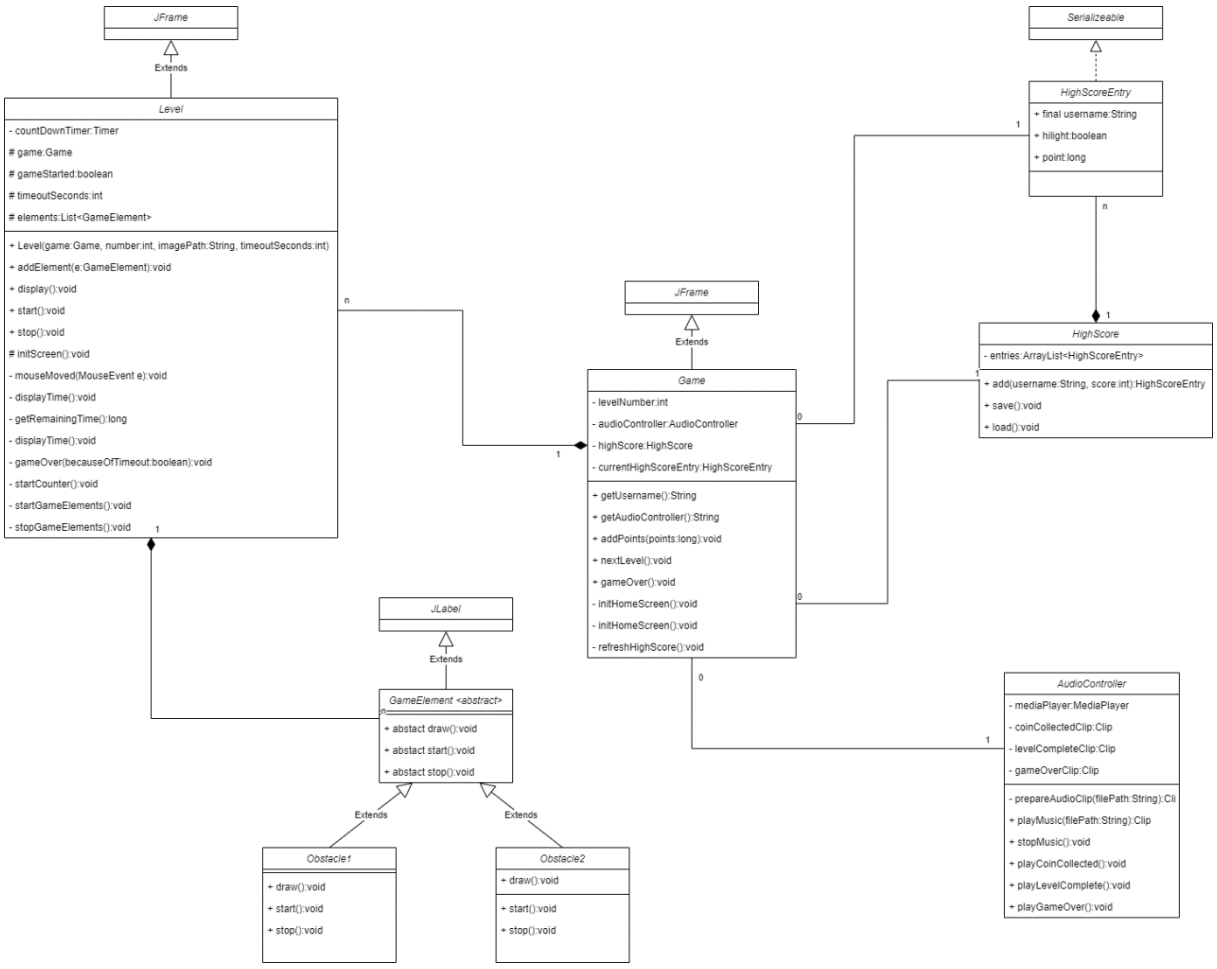
- Die Farbe der Maus-Position wird laufend gelesen, wenn sie während des Spiels nicht weiß ist, ist das Spiel zu Ende. Das funktioniert auch mit den Animated-Gifs, welche sich über den Bildschirm bewegen.
- Die Punkte werden in einer Arraylist gesammelt, absteigend sortiert und in einer Tabelle am Monitor dargestellt.

Mockup



Architektur

UML Klassendiagramm



Klassenbeschreibungen

- Zentrale Klasse ‚Game'
  Die zentrale Klasse wird aus JFrame abgeleitet. Hier findet die Benutzerabfrage statt und es wird zu anderen Klassen wie Highscore und Audiocontroller referenziert. Die Levels werden erzeugt, organisiert und gestartet. Auch das Logfile wird hier initialisiert.

- Klasse ‚Level'
  Diese Klasse wird ebenfalls aus JFrame abgeleitet und hat die Fähigkeit ein Hintergrundbild darzustellen. Hier ist die Spielelogik mit Start und Ende implementiert sowie die Farberkennung bei Maus-Bewegung. Der Timer für den Countdown ist in dieser Klasse enthalten und Hindernisse (Game Elements) können hinzugefügt werden.

- Klasse ‚GameElement'
  Es handelt sich um eine abstrakte Klasst mit der Fähigkeit Animated-Gifs darzustellen. Zu den Levels hinzugefügt werden sie in der Game-Klasse. Davon abgeleitete Klassen sind ‚Obstacle1' und ‚Obstacle'.

- Klasse ‚Highscore'
  In dieser Klasse werden die Highscore-Objekte in einer Arraylist gesammelt. Die Arraylist kann serialisiert werden, in ein File gespeichert werden und das File wieder geladen und deserialisiert werden.

- Klasse ‚Highscore-Entry'
  Die Klasse definiert die Einträge der Highscore-Tabelle, nämlich User*in-Name, erreichte Punktezahl und der aktuellste Eintrag wird in einer anderen Farbe (rot) dargestellt. Die Liste wird absteigend nach Score sortiert.

- Klasse ‚AudioController'
  Mit dieser Klasse kann Hintergrundmusik und kurze Clips bzw. weitere Animated-Gifs eingespielt werden.

Logging

Für das Logging wurde ‚Log4j' verwendet. Das Logging wird in der Game-Klasse initialisiert und ist von überall durch eine final static-Variable zugänglich. Die Konfiguration ist in den log4j-properties enthalten. Die Logdateien inkl. Logrotation werden im Root-Verzeichnis des Spiels gespeichert.

Unit Tests

Dafür wurde ‚JUnit' verwendet, das VSCode-Plugin ‚Testrunner for Java' war zusätzlich erforderlich. Im ersten Test wird ein Highscore erstellt und mit dem zu erwartenden HTML verglichen. Wenn der Vergleich positiv ausfällt, ist der Test erfolgreich.
Im zweiten Test wird ein Highscore mit 10.000 Einträgen erstellt und gespeichert. Wenn der Highscore nach dem Laden gleich ist wie vor dem Speicher, ist der Test erfolgreich.

Dokumentation

Das Klassendiagramm wurde mit draw.io erstellt. Detaillierte Kommentare wurden direkt im Code eingefügt. Zusätzlich wurden alle Methoden kommentiert und daraus wurde ein Java-doc-Html generiert.

Abschluss

Das Spiel ist fertig programmiert (Erweiterungen z.B. weitere Levels und Münzen sammeln ist möglich), wurde ausführlich dokumentiert und getestet und kann gestartet werden!

Viel Spaß!