# Software Entwicklung 2

*Wintersemester 2020/21*

*MI7*


# PixelJumper

**Entwickler**

Cazim Ukela[cu011@hdm-stuttgat.de]

Daniel Hiller [dh102@hdm-stuttgart.de]

Niklas Mäckle [nm067@hdm-stuttgart.de]

https://gitlab.mi.hdm-stuttgart.de/nm067/se2-projekt.git

# Dokumentation

## Kurzbeschreibung

Bei unserem Projekt handelt es sich um ein tilebasiertes Singleplayer Game, mit der Möglichkeit eigene Maps zu erstellen und Highscores zu erzielen. Es handelt sich um ein Jump & Run -Game, welches sich im Arcade Genre einreiht.

Im Map-Editor stehen diverse Tiles zur verfügung, welche man durch einen Klick auswählen kann und dann entweder durch einen weiteren Klick oder Drag in der Map platzieren kann. Wenn man mit dem erstellen der Map fertig ist, hat man die Möglichkeit diese über die GUI zu exportieren, welche dann in Form einer .json-Datei persistent gemacht werden.

Wenn man das Spiel startet, kann man eine Map auswählen, auf welcher man dann an der Startflagge spawnt. Sobald man gespawnt wird läuft ein Timer, welcher erst stoppt, wenn man das Ziel erreicht. Auf dem Weg dahin versucht man so viel Münzen wie möglich einzusammeln, da sich der Score aus den gesammelten Münzen und der benötigten Zeit berechnet. Sobald man das Ziel erreicht hat, erscheint ein Fenster in welchem man ein Pseudonym eintragen kann, unter welchem der Score gespeichert wird.

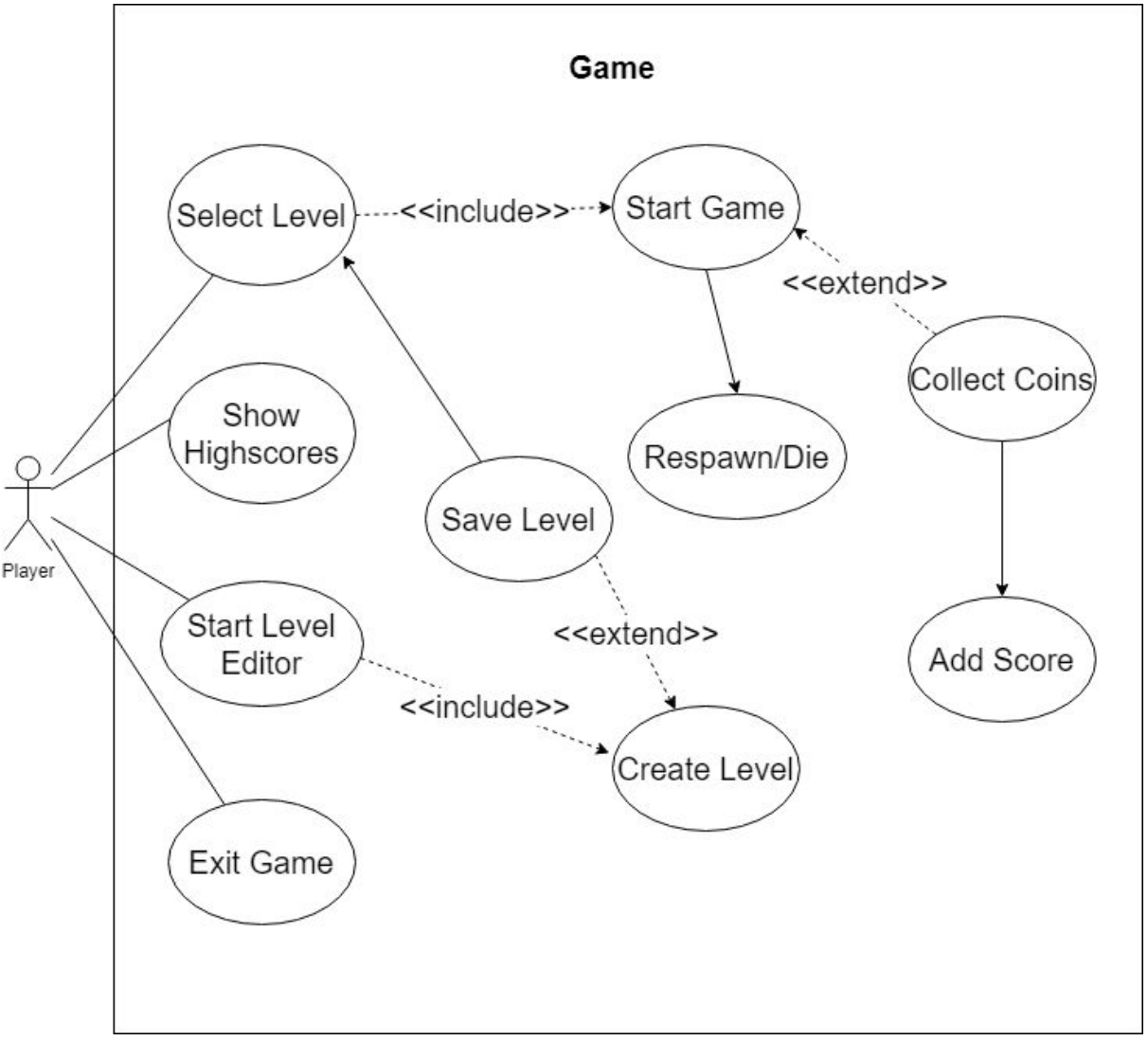
Beim erstellen einer Map wird für diese eine .dat-Datei angelegt, in der alle Scores, welche auf der Map erzielt wurden, gespeichert werden. Im Hauptmenü existiert eine eigene GUI, in welcher man sich die Maps und die auf ihr erzielten Highscores anzeigen lassen kann.

## Startklasse
Die Main-Methode befindet sich in der Klasse **Main**

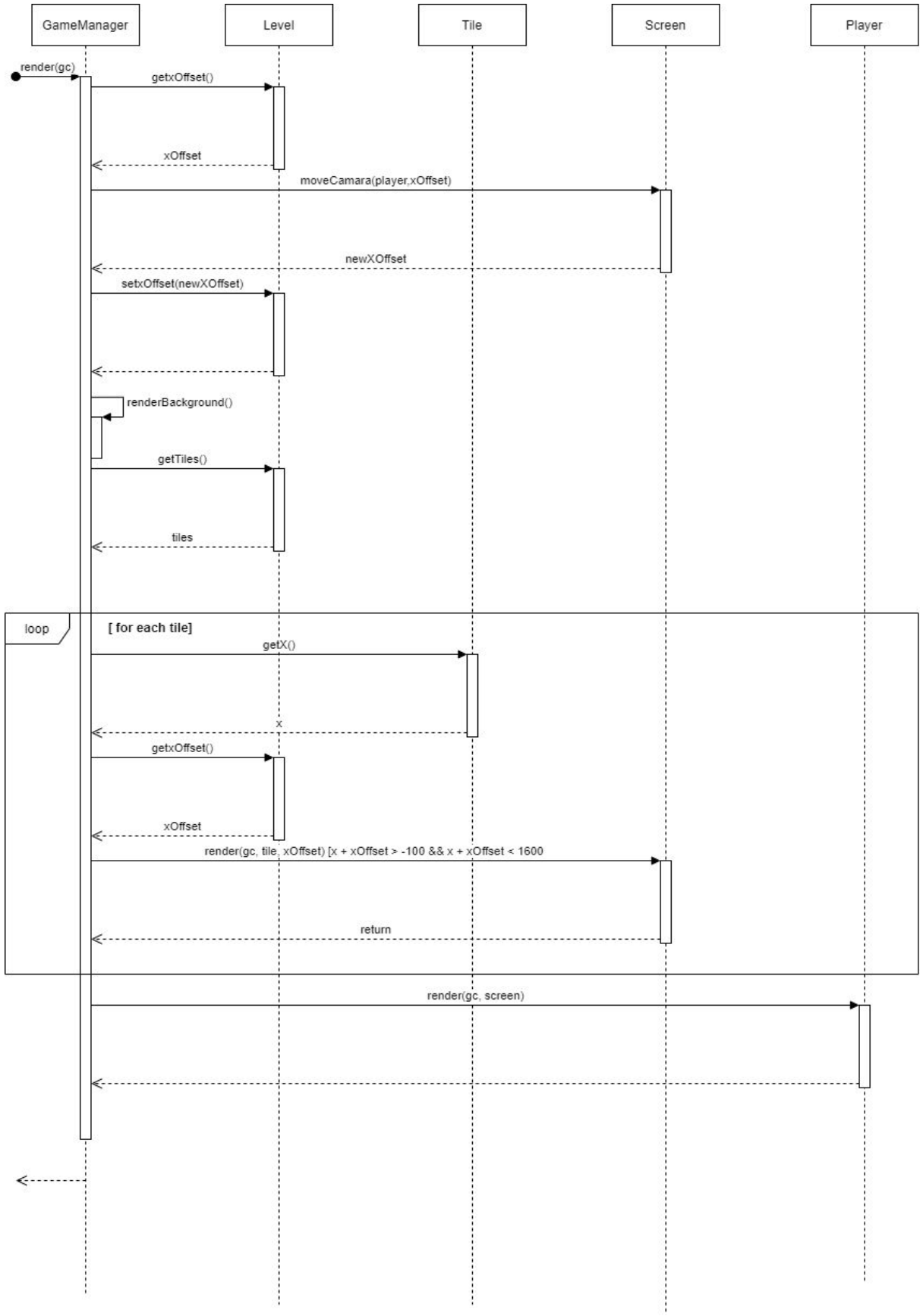
## Besonderheiten
Aktuell ist es noch so, dass auch das Wasser tödlich ist. Dies könnte man in einem späteren Update verändern. Unter anderem funktionieren die Leitern noch nicht so wie gewünscht, daher sind die Leitern aktuell nicht in funktion. Es wurde die JDK 14.0.2 verwendet.

## UML-Diagramm
Klassendiagramm: Das Klassendiagramm passt auf Grund der größe nicht in die Dokumentation, dies bitte dem Repository entnehmen (bitte mit draw.io öffnen)

Use Case Diagramm:



Sequenzdiagramm:



## Stellungnahme

Architektur:
(Erweiterbare Architektur mit Interfaces und Factories)

Im Projekt gibt es das Interface **ITile**[src/main/java/de/hdm_stuttgart/mi/se2/game/level/tiles] welches einfach erweiterbar ist. Unter anderem benutzen wir zur erstellung der Tiles eine **TileFactory**[src/main/java/de/hdm_stuttgart/mi/se2/game/level/tiles]. Diese wird benutz um einfach neue Tile mit bestimmten Attributen zu erstellen. Dies ist eben so durch den switch case einfach erweiterbar.

Clean Code:
Es werden Getter sowie Setter verwendet. Siehe z.B. Klasse [src/main/java/de/hdm_stuttgart/mi/se2/game/level/] **Level**

Dokumentation und Tests:
Alle Klassen, sowie wichten Methoden wurden dokumentiert.
Siehe Package [src/test/java/de/hdm_stuttgart/mi/se2/game]

GUI (JavaFX):
(Komplexe GUI vorhanden)
Wenn das Programm gestartet wird, gelangt man auf das Main menü. Dieses besitz wiederum Buttons um das Spiel zu starten oder in den Leveleditor zu gehen. Unter anderem wird beim speichern des erstellten Levels ein neues Fenster geöffnet um den Levelnamen eingeben zu können.

Logging/Exceptions:
(Logging mit Logging Framwork)
Wir verwenden das Logging Framwork von **apache**(log4j). Verwendet z.B. in der Klasse **Level**[src/main/java/de/hdm_stuttgart/mi/se2/game/level/] in der Methode **loadLevel()**

(Sinnvolle Anwendung von Logstufen)
Wie schon im Beispiel von oben gezeigt wird einmal die Logstufe **info** verwendet. In der Methode gibt es aber auch eine Exception die geloggt wird. Diesmal aber mit der Logstufe **error**.

(Logging von Exceptions und Threads)
Das Logging von Exceptions wurde schon im Punkt davor gezeigt. Das Logging von Threads findet in der Klasse GameController statt.

UML:
Klassendiagramme, Use-Case Diagramm und Sequenzdiagramm vorhanden und eigenständig erstellt

Threads:
In der Klasse [src/main/java/de/hdm_stuttgart/mi/se2/game/controller]**GameController** wird ein Thread erstellt, welcher ein Musikstück abspielt. Dieser wird gestartet, wenn man ein Level zum spielen ausgesucht hat und wird gestoppt wenn man das level fertig gespielt hat.

Streams:
(Streams parallel verarbeiten, ohne Race Condition)
Die Verarbeitung findet in der Klasse **Map** [src/main/java/de/hdm_stuttgart/mi/se2/game/level/map] in der Methode **getMapNamesAsArray()** statt.

Factories:
Unsere Factory befindet sich in der Klasse **TileFactory**[src/main/java/de/hdm_stuttgart/mi/se2/game/level/tiles].

## Bewertungsbogen

| Vorname | Nachname | Kürzel | Matrikelnummer | Projekt | Arc. | Clean Code | Doku | Tests | GUI | Logging/Except. | UML | Threads | Streams | Profiling | Summe - Projekt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Daniel | Hiller | dh102 | 40181 | PixelJumper | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 30.00 |
| Niklas | Mäckle | nm067 | 40060 | PixelJumper | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 30.00 |
| Cazim | Ukela | cu011 | 40064 | PixelJumper | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 30.00 |