A screen shot of a cellphone

Description automatically generated

School Skipper

DoNnErB GmbH

**Inhaltsverzeichnis**

[1. Einleitung 3](#_Toc161087328)

[2. Spielziel 3](#_Toc161087329)

[3. Architekturentwurf 3](#_Toc161087330)

[4. Projektdurchführung 3](#_Toc161087331)

[4.1. Implementierung 3](#_Toc161087332)

[4.1.1. Benutzeroberfläche 3](#_Toc161087333)

[4.1.2. Implementierte Klassen 4](#_Toc161087334)

[5. Marketingstrategie 5](#_Toc161087335)

[6. Schluss 5](#_Toc161087336)

[6.1. Übergabe 5](#_Toc161087337)

[6.2. Fazit 5](#_Toc161087338)

# Einleitung

Projektdokumentation des „Doodle Jump“ Klons „School Skipper“ mit Online- und Chat-Funktion.

# Spielziel

Ziel des Spiels ist es von einer Plattform zur nächsthöheren zu springen und dadurch höher als der Gegner zu kommen. Auf den Plattformen können Gegner stehen, die bei Kontakt mit dem Spieler ein sofortiges Ausscheiden zur Folge haben. Gewinner ist der Spieler, der am höchsten springt bzw. die höhere Punktzahl erreicht.

# Architekturentwurf

**Siehe UML-Klassendiagramme im Teams-Folder**

# Projektdurchführung

Im Folgenden Abschnitt wird die gesamte Projektdurchführung dargestellt und genauer beschrieben.

# Implementierung

# Benutzeroberfläche

Für die Benutzeroberfläche wurde nicht auf Qt Formulardateien gesetzt, sondern programmatisch das gesamte Layout definiert. Auch wurde auf ein dynamisches Layout gesetzt, so dass der Nutzer die Fenstergröße nach seinen Präferenzen einstellen kann und sich die Widgets automatisch anpassen. Das gesamte Design wurde schlicht und intuitiv für den Benutzer gestaltet, daher finden sich klare Symbole und Labels auf allen Knöpfen im gesamten User Interface.

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedUser Interface mit Chat eingeklappt**

**User Interface mit ausgeklapptem Chat**

# Server

**Netzwerk**

Der Server ist mit den Clients in einer UDP-Multicastgruppe umso die Spielerkoordinaten, und damit die Spielzüge zu dokumentieren. Des Weiteren hört der Server auf einen TCP-Port, an den die Clients Nachrichten schicken können und diese so dem anderen Spieler mitteilen. Ebenfalls wird über den TCP-Port für das Spiel wichtige Steuerinformationen übermittelt z.B., wenn ein Spieler aus dem Spiel ausscheidet, sendet dieser die Nachricht „Dead“.

**CSV-Erstellung**

Der Server legt bei Spiel beginn eine Datei an, deren Name die „Datetime“ ist. In dieser Datei speichert der Server alle ihm bekannten Spielzüge ab. Der Server legt diese im Format: IP-Adresse des Senders + den Szenenkoordinaten an. Sollte einer der Clients ausscheiden stoppt der Server die weitere Aufzeichnung der Nachrichten und beendet die Datei mit dem Eintrag „Game Over“.

**Besonderheiten**

Der Server erbt von der Klasse QObject, um Zugriff auf die Signal-Slot Funktionalität von QT zu haben. Des Weiteren verwendet die Klasse intern die QT-Klassen „TcpServer“ und „UdpSocket“ für die Netzwerkverbindungen. Der Server ist gemäß den Anforderungen als Singleton-Pattern implementiert.

# Client

**Netzwerk**

Der Client tritt einer Multicastgruppe, um so die Koordinaten des anderen Spielers zu erhalten, sowie die eigenen Koordinaten zu verschicken. Für die Kommunikation mit dem anderen Spieler, wird eine TCP-Verbindung mit dem Server hergestellt, um die Vollständigkeit der Nachrichten zu gewährleisten.

**Spiel**

Das Spiel selbst ist in mehrere Klassen unterteilt. Die Game Klasse, die Actor Klasse und die Plattform Klasse. Die Klassen Actor und Plattform erben die Funktionalität der Signale und Slots aus QObject, während Sie die Darstellungsoptionen und Kollisionsberechnungen von QGraphicsPixmapItem erben. In den Klassen wurden ebenfalls die Kollidierenden Bereiche angepasst, sodass diese nicht die gesamte Grundfläche der Bilddateien einnimmt. Das Spiel selbst basiert auf einer einfachen Schleife, die in Abhängigkeit von einem QTimer aufgerufen wird. Innerhalb dieser Schleife werden mögliche Kollisionen geprüft, um anschließend den Spieler zu bewegen. Sollte der Spieler zu weit nach unten abstürzen, verliert er das Spiel.

**GUI**

Eine Graphische Oberfläche, um das Spiel entweder im Multiplayer oder Singleplayer zu starten, sowie ein Chatfenster, um Nachrichten an den anderen Spieler zu versenden. Das GUI besteht aus den eigenen Klassen SchoolSkipperClient und CustomButton, sowie aus den QT-Klassen des QPushButtons, QLabel und QTextBrowser.

# Implementierte Klassen

**Siehe Doxygen Doku (annotated.html) im Teams-Folder**

# Marketingstrategie

**Siehe Marketing-Strategie im Teams-Folder**

# Schluss

# Übergabe

Das Projekt wird durch eine Projektpräsentation der Klasse vorgestellt und der gesamte Source Code ist in einem öffentlichen GitHub zum Download verfügbar.

# Fazit

Durch das Projekt konnten wir unser Wissen über die Qt-Bibliothek erweitern und festigen. Negativ war die mangelnde Zeit, die vor allem zu Lasten einiger Features ging. Eine längere Projektdauer würde hauptsächlich der Server Implementierung zugutekommen, die bei vielen Projektgruppen nicht komplett umgesetzt werden konnte. Positiv war die freie Auswahl und Gestaltung der Spiele.