Hochschule für Technik und Wirtschaft Berlin

Fachbereich 1:
Computer Engineering

Softwaretechnik Projekt

Super Mario

Dozent:

Prof. Dr. Sebastian Bauer

Ali Al Ali

Mohamed Salim Mhaamdi

Ilyess Sallemi

Sommersemester 2020/2021

Inhaltverzeichnis:

1. Einführung 2
2. QT Beschreibung 2
3. Software Beschreibung 3
4. Aufbau des Programmes.....4
5. Modellierung 5
6. Klassendiagramm.....6
7. Beschreibung der Implementierung:.....6
    - Erste Erweiterung.
    - Zweite Erweiterung.
    - Dritte Erweiterung.

7.Fazit.....11

Abbildungsverzeichnis:

Abbildung 1: Screen Schot das Spiel Super Mario mit QT 3

Abbildung 2: Sourcetrail Übersicht 4

Abbildung 3: Use-Case-Diagramm 5

Abbildung 4: Klassendiagramm Source-Code 6

Abbildung 5: Mit V Taste fliegt Mario 7

Abbildung 6: 5 Leben für Mario hinzufügen 8

Abbildung 7: Fuzzy 8

1. Einführung:

Softwareentwicklung ist ein „großes“ Wort und beinhaltet so viel mehr als nur Programmieren. Während unserem Projekt im Modul Softwaretechnik beschäftigen wir uns mit einer Erweiterung in einer bestimmten Software. In unserem Fall haben wir uns entschieden für das Spiel Super Mario.
Im Kontext der Entwicklung einer Software lernen wir nicht nur wie eine Software funktioniert, sondern auch wie wir unsere Kenntnisse in Programmierung benutzen.

2. QT Beschreibung:

QT ist ein Anwendungsframework und GUI-Toolkit zur plattformübergreifenden Entwicklung von Programmen und grafischen Benutzeroberflächen. Darüber hinaus bietet Qt umfangreiche Funktionen zur Internationalisierung sowie Datenbankfunktionen und XML-Unterstützung an und ist für eine große Zahl an Betriebssystemen bzw. Grafikplattformen wie X11 (Unix-Derivate), MacOS, Windows, iOS und Android erhältlich. Qt wird insbesondere vom KDE-Projekt in den Bibliotheken der KDE Plasma Workspace und der KDE Frameworks verwendet, die gleichzeitig die prominentesten Vorzeigebeispiele darstellen.

3. Software Beschreibung:

Mario ist die populärste Videospiel-Figur der Firma Nintendo und deren Maskottchen.
Bei Mario handelt es sich um einen schnauzbärtigen, etwas klein gewachsenen und dicklichen italienischen Klempner mit blauer Latzhose, rotem Hemd und roter Schirmmütze mit einem M-Symbol, dessen typische Aussprüche „Mamma mia“ und „It’s-a-me, Mario“ sind.

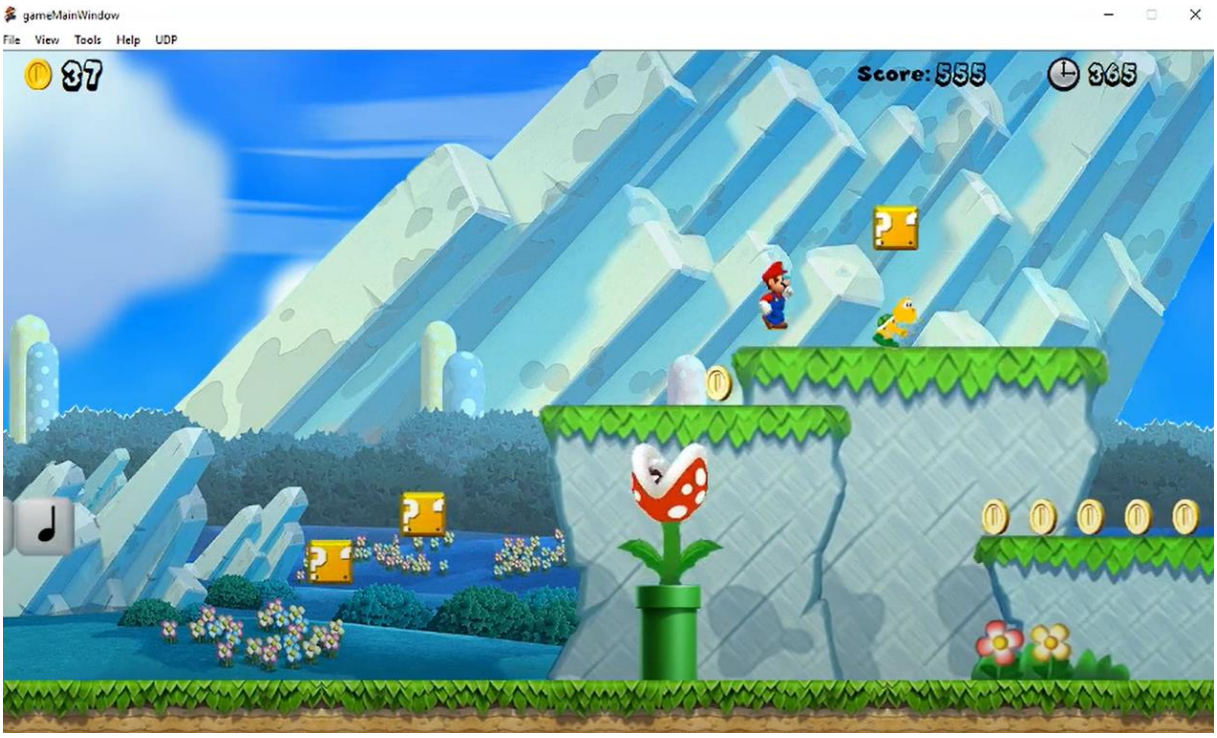

Abbildung 1: Screen Schot das Spiel Super Mario mit QT

4. Aufbau des Programmes:

- Programmiersprache:

  Super Mario ist ein Jump-'n'-Run-Videospiel des japanischen Unternehmens Nintendo. Das Spiel wurde in der Sprache C++ programmiert.

  Während unserem Projekt werden wir QT um die Bearbeitung und die Ausführung von unserem Spiel benutzen.

- Übersicht des Quellcodes:

  Zum Anfang des Projekts, wurden von uns die Analyse-Tools Sourcetrail und Doxygen genutzt, um eine Übersicht vom Umfang und Beschaffenheit des Quellcodes zu bekommen.

  Sourcetrail erwies sich hierbei als komfortabler und übersichtlicher. Wir konnten mit Hilfe dieses Tools einfacher den Aufbau und die Beschaffenheit der Klassen und Funktionen erfassen.

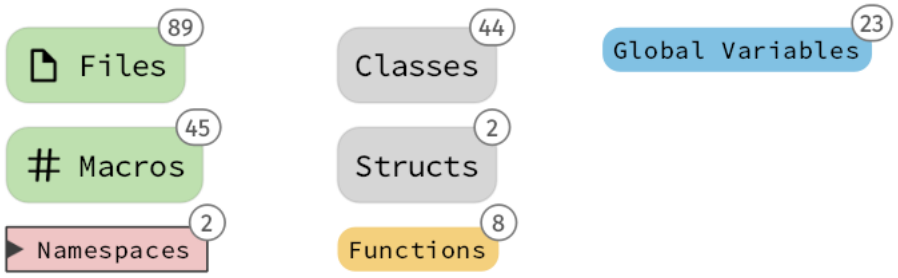

Abbildung 2: Sourcetrail Übersicht

Insgesamt umfasst der Code 7456 Zeilen an Code in 89 Dateien. Durch die Analyse mit Sourcetrail konnten wir uns einen Überblick, über die wichtigste Ursprungsklasse von der alle GUI Elemente abhängen verschaffen.

5. Modellierung:

Use-Case-Diagramm:

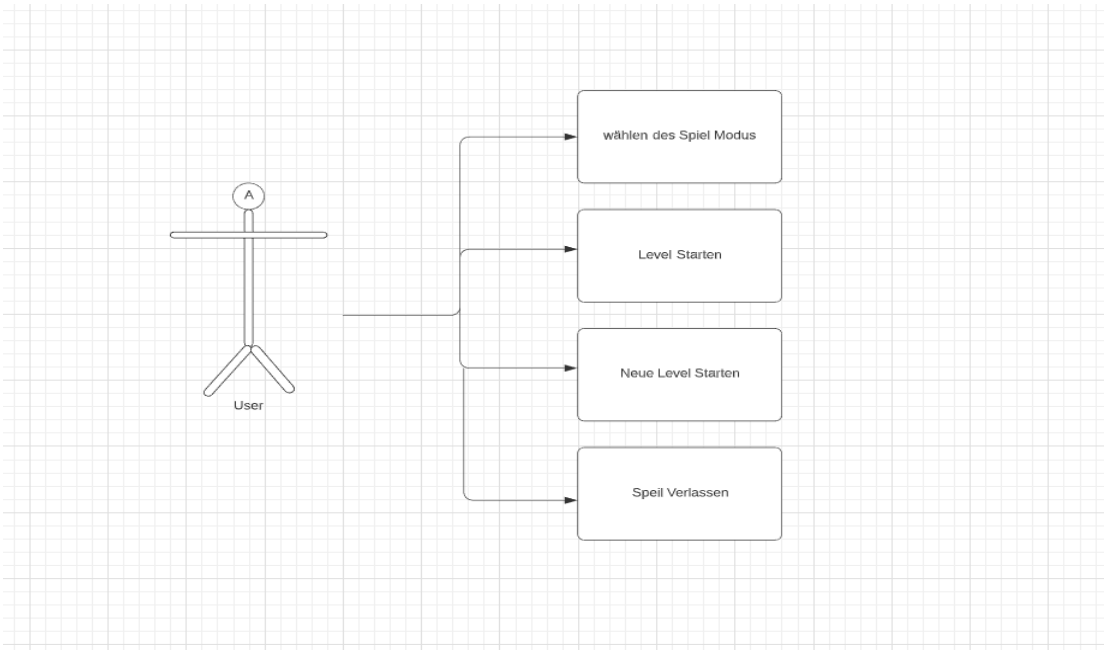

Abbildung 3: Use-Case-Diagramm

Use Case Modell:

Vorbedingung: In jedem Level zeigt das System ein oder mehrere Coins an.
Nachbedingung: Der Spieler soll, wenn möglich alle Coins einsammeln um höhere Level anzufangen.

- Wählen des Spiel Modus:

Das System zeigt die Modus Guest oder Login an.

Der Spieler wählt den Modus

- Level Starten:

Mario überholt alle Feinde.

Mario sammelt alle Coins.

Mario übertrifft das Level.

- Neue Level Starten:

Wenn Mario das Spiel nicht erfolgreich abschließt.

- Speil Verlassen:

Use Case endet erfolgreich.

5. Klassendiagramm

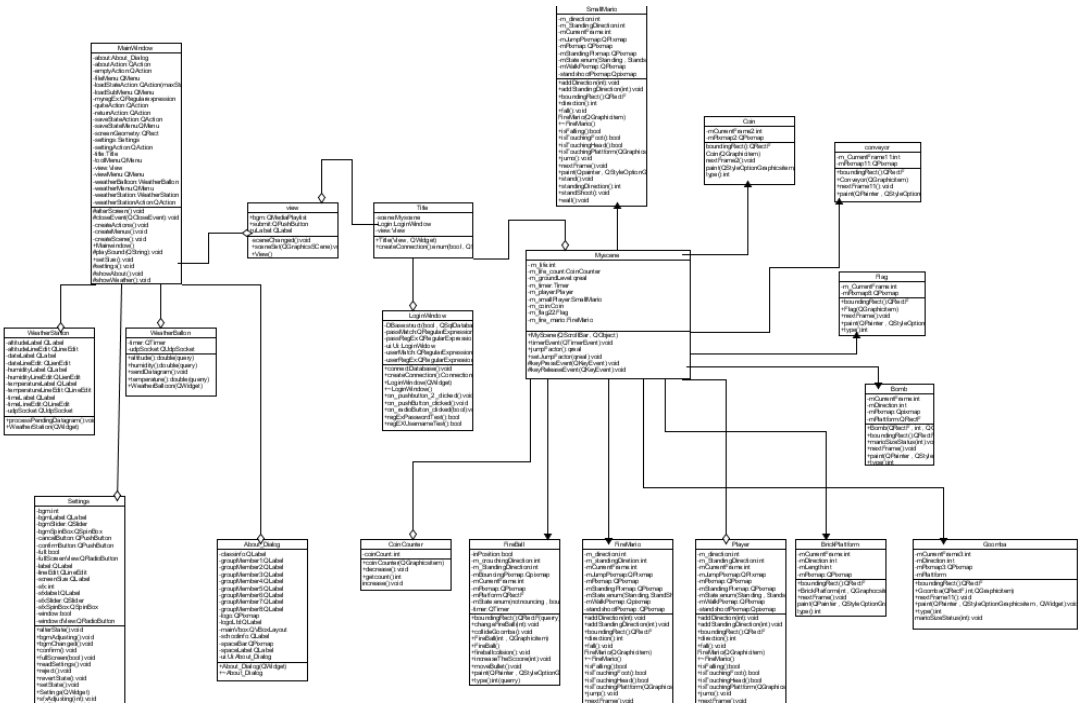

Abbildung 4: Klassendiagramm Source-Code

6. Beschreibung der Implementierung:

An der Software sollen nach Absprache mit dem Dozent Modifikationen im Code beziehungsweise in der GUI vorgenommen werden. Diese Änderungen werden im Folgenden näher erläutert.

- Erste Erweiterung:

Wenn man auf V drückt fliegt Mario 5 Sekunden.

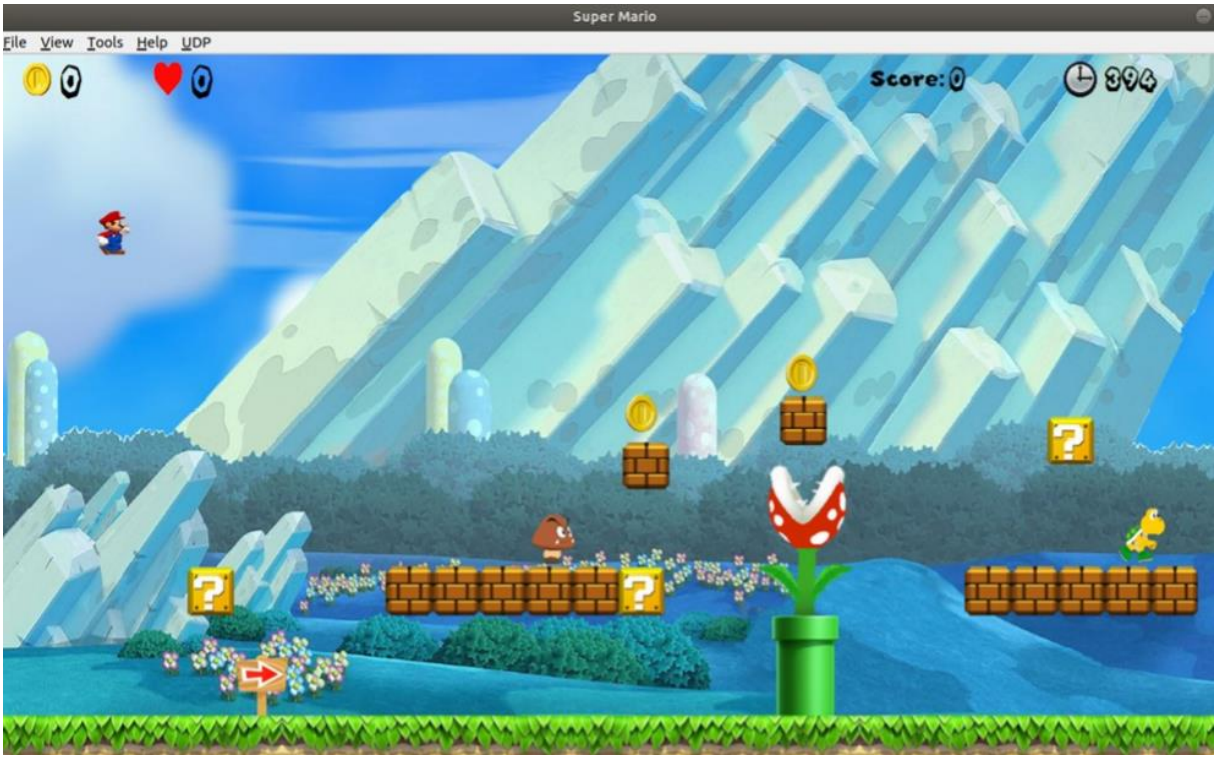

Abbildung 5: Mit V Taste fliegt Mario

- Zweite Erweiterung:

Ein lebensgegenstand für das Spiel hinzufügen.

Ein Herz Bild wird gezeigt auf dem Bildschirm ganz oben links, genau neben der Coinzähler.

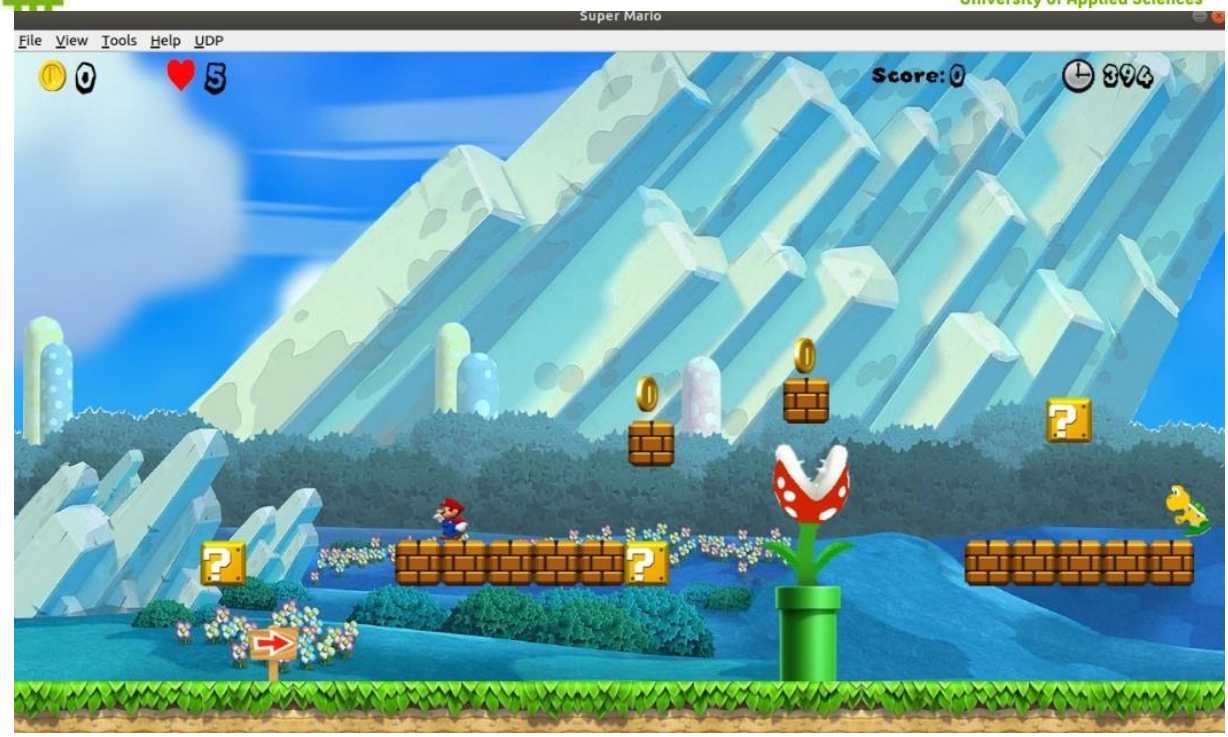

Abbildung 6: 5 Leben für Mario hinzufügen

- Dritte Erweiterung:

Neuen Feind hinzufügen Fuzzy:

Eine neue Klasse wird erstellt mit dem Namen Fuzzy:

Abbildung 7: Fuzzy

Der Header File:

```
#ifndef Fuzzy_H
#define Fuzzy_H
#include <QGraphicsItem>
#include <QPixmap>
#include "objecttype.h"
#include <QObject>

class MyScene;
class CoinCounter;

class Fuzzy : public QObject,public QGraphicsItem
{
   Q_OBJECT
public:
   enum { Type = UserType + FuzzyType };
   Fuzzy(QRectF platformRect, int direction,QGraphicsItem *parent = 0);
   QRectF boundingRect() const;
   void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget
     *widget);
   int type() const;
public slots:
   void nextFrame();
signals:
   void marioSizeStatus(int);
private:

   int mCurrentFrame;
   QPixmap mPixmap;
   QRectF mPlatform;
   int mDirection;
};
#endif // Fuzzy_H
```

Der CPP File:

```
#include "Fuzzy.h"
#include <QPainter>
#include "myscene.h"
#include "coincounter.h"

Fuzzy::Fuzzy(QRectF platformRect, int direction,QGraphicsItem *parent)
   : QGraphicsItem(parent)
     ,mCurrentFrame(),mPlatform(platformRect), mDirection(direction)

{
   setFlag(ItemClipsToShape);
   mPixmap = QPixmap(":images/redt.png");
   QTimer *timer = new QTimer(this);
   connect(timer,SIGNAL(timeout()),this,SLOT(nextFrame()));
   timer->start(100);
}
```

```
void Fuzzy::nextFrame(){

    mCurrentFrame += 179;
    if (mCurrentFrame >= 3521 ) {
        mCurrentFrame = 0;
    }
    if(this->pos().x() < mPlatform.x() || this->pos().x() > mPlatform.x()+ mPlatform.width())
     {
        mDirection = -mDirection;
        setTransform(QTransform(-mDirection, 0, 0, 1, boundingRect().width(), 0));
    }
    setPos(this->pos().x() + (mDirection*7), this->pos().y());

    //Collision Detection
        QList<QGraphicsItem *> colliding_items = collidingItems();

        //If one of the colliding items is an Enemy, destroy both the bullet and the enemy
        for (int i = 0, n = colliding_items.size(); i < n; ++i){
            if (typeid(*(colliding_items[i])) == typeid(Player)){
                g_nFlag = 0;
                emit marioSizeStatus(0);
                delete colliding_items[i];
                qDebug()<<"Mario deleted small mario added";
                delete this;
                return;
            }
        }
        for (int i = 0, n = colliding_items.size(); i < n; ++i){
            if (typeid(*(colliding_items[i])) == typeid(SmallMario)){

                // emit marioSizeStatus(1);
               //  delete colliding_items[i];
                // qDebug()<<"small mario dead";
                 g_life_count->decrease();
                 delete this;
                 return;
            }
        }
        for (int i = 0, n = colliding_items.size(); i < n; ++i){
            if (typeid(*(colliding_items[i])) == typeid(FireMario)){

                // emit marioSizeStatus(6);
                 //delete colliding_items[i];
                // qDebug()<<"Fire mario deleted small mario added";
                 delete this;
                 return;
            }
        }
}

QRectF Fuzzy::boundingRect() const {
    return QRectF(0,0,130,146);
```

```
    }

    void Fuzzy::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget
        *widget) {
        painter->drawPixmap(0,0, mPixmap, mCurrentFrame, 0,130, 146);
        setTransformOriginPoint(boundingRect().center());
        Q_UNUSED(widget);
        Q_UNUSED(option);

    }

    int Fuzzy::type() const {
        return Type;
}
```

7. Fazit:

Wir hatten viele Schwierigkeiten bei der Implementierung des Codes von Super Mario. Zuerst mussten wir verstehen, wie QT funktioniert und wie wir den benutzen können, um die Veränderung zu schreiben. Der Code war nicht zu schwer zu verstehen, da wir schön mit der Programmiersprache C++ viele zu tun hatten. Viele andere Schwierigkeiten traten als wir unserem Code compiliert hatten.

Wir haben schön viel gelernt, wie wir zusammen in einer Gruppe die Probleme lösen. Wir haben auch Erfahrung gesammelt, ein code von git repository herunterzuladen, öffnen und Erweiterung zu schreiben.