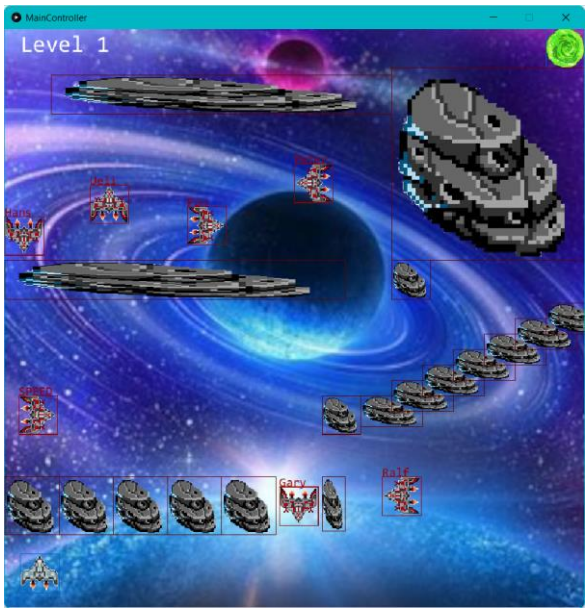
DOKUMENTATION **«Space Ship»**

Abschlussprojekt Modul 226b



Projektverfasser: **Wendelin Haller**

Ausbildung:	Informatiker / Applikationsentwickler
Schule:	WISS St. Gallen, 3. Semester 2021/2022
Lehrperson:	Sven Schirmer
Modulbezeichnung:	Objektorientiert implementieren

Abgabetermin: 25. Januar 2022

Inhalt

EINLEITUNG

Im 3. Semester meiner Ausbildung zum Informatiker-Applikationsentwickler and der WISS (Schulen für Wirtschaft Informatik Immobilien) St. Gallen habe ich denn Auftrag bekommen ein Abschlussprojekt für das Modul 226b (Objektorientiert implementieren) zu machen und das hier ist die Dokumentation zum ganzen Projekt. Für die Projektarbeit standen uns zehn Wochen Zeit vom 22.11.2021 bis 25.01.2022 zur Verfügung.

1. PROJEKT «Space Ship»

Ein Einzelspieler Spiel mit zwei Levels habe ich programmiert. In diesem Spiel kann man mit den Tasten ‘W’ , ‘A’, ‘S’ und ‘D’ ein Raumschiff in alle vier Himmelsrichtungen bewegen. Das Ziel ist es das Raumschiff sicher und unversehrt zum grünen End Portal zu manövrieren. Auf dem Weg sind aber noch Gegner, die eine strickte Route abfliegen und den Spieler (Das Raumschiff) bei Berührung ausser Gefecht setzen. Die Gegner können auch ausser Gefecht gesetzt werden, indem man mit dem Raumschiff auf sie zufliegt und mit der Maus klickt. Dann wird ein kleines grünes Projektil abgeschossen. Wenn man den Gegner damit trifft, ist er ausser Gefecht gesetzt. Dabei werden alle anderen Gegner alarmiert und rasten aus, indem sie schneller fliegen und anfangen zu schiessen. Das dauert nicht lange und sie beruhigen sich wieder. Nicht nur Gegner sind dir im Weg, sondern auch sich nicht bewegende Asteroiden-Wände. Diese setzen dich auch ausser Gefecht, wenn du sie berührst.

2. INFORMIEREN

Im Verlaufe des Projekts sind Mehrer Probleme und Schwierigkeiten aufgetaucht und die folgende Liste zeigt auf, was für Probleme aufgetaucht sind, mit welchen für Hilfestellungen ich dieselösen konnte und wo ich Infos geholt habe.

Probleme	Wie ich das Ganze auf GitHub packen soll.
Hilfestellung	https://www.w3schools.io/file/markdown-list/
Probleme	Wie soll ich das Status Controls machen? und habe dann alle keyPressed Methoden in die MainController gepackt und den View in die einzelnen View-Klassen.
Hilfestellung	Ich und meine Gedanken.
Probleme	Image für das Raumschiff einbinden.
Hilfestellung	https://discourse.processing.org/t/applying-a-background-image-to-a-rect/13711/3
Probleme	Nicht gewusst wie anfangen und was ich genau die nächsten Male machen soll.
Hilfestellung	https://templates.office.com/en-us/Simple-Gantt-Chart-TM16400962
Probleme	Darstellung der Asteroiden.
Hilfestellung	- https://www.ecosia.org/images?q=bixle%20art%20asteoride - https://stackoverflow.com/questions/36511675/i-cant-use-image-and-rect-in-the-draw-method-in-javascriptmode - https://py.processing.org/reference/noFill.html
Probleme	java.util.ConcurrentModificationException
Hilfestellung	Sven Schirmer
Probleme	Langsames Game wegen Bilder, die immer im Draw geladen werden.
Hilfestellung	Sven Schirmer
Probleme	Wie erstellt man Mockup Test?
Hilfestellung	Modul Präsentationen und Beispiele wo ich schon Mockup-Tests erstellt habe.
Problem e	Auswal der Klassen und was getestet werden soll
Hilfestellung	Sven Schirmer

3. Planen

Ich habe das Projekt in drei Themen unterteilt, diese sind: «Start», «Implementieren», «Testen und Verbessern». Jedem Thema habe ich eine Farbe zugeteilt und diese im Zeitplan verwendet. *(siehe Abbildung 1)*

Der grobe Zeitplan meines Projektes war, dass ich zuerst ein Gerüst programmiere mit Start, Level und End-View. Um danach das eigentliche Spiel und die Funktionalität in den Level-View einzubauen, damit ich nicht, wenn ich das Spiel fertig habe, noch einen Start und End-View drumherum bauen muss. Nachdem dann die Views und das Spiel und dessen Funktionalitäten ausprogrammiert habe, setzte ich mich daran, drei Klassen zu testen mit Unit und Mockup-Tests. Zu guter Letzt, habe ich noch diese Dokumentation eingeplant.

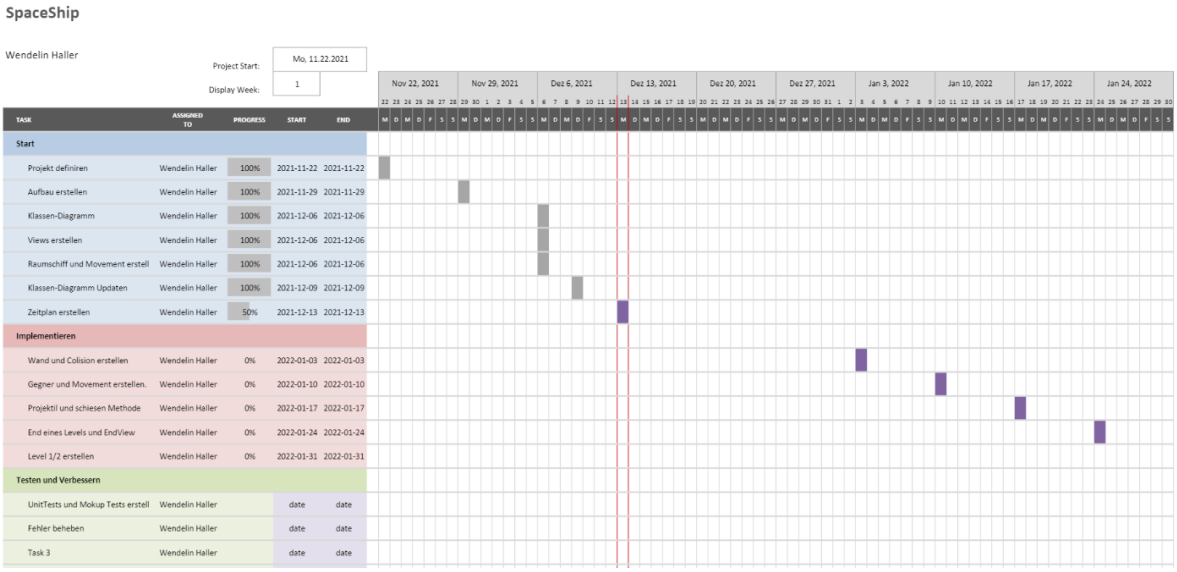


Abbildung 1 : Zeitplan 13.12.2021 (siehe vergrösserte Version im Anhang.)

Als ich angefangen am 13. Dezember 2021 hatte mit dem ersten Zeitplan da habe ich noch ein paar Schritte übersehen und den Zeitaufwand unterschätzt. Deswegen habe ich am 21. Januar 2022 einen neuen Zeitplan erstellt und die drei Themen zu Start/Implementieren, Testen und Verbessern und Dokumentation verändert und ein paar Tasks anders geplant und neue erstellt. *(siehe Abbildung 2)*

Das Einschätzen, wie lange ich für einen Task brauche, habe ich immer schwierig gefunden und habe meist länger gebraucht, als gedacht.

SpaceShip

Wendelin Haller

Project Start: Mo, 11.22.2021

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Start / Immplementieren				
Projekt definiren	Wendelin Haller	100%	2021-11-22	2021-11-22
Aufbau erstellen	Wendelin Haller	100%	2021-11-29	2021-11-29
Klassen-Diagramm	Wendelin Haller	100%	2021-12-06	2021-12-06
View's erstellen	Wendelin Haller	100%	2021-12-06	2021-12-06
Raumschiff und Movement erstell	Wendelin Haller	100%	2021-12-06	2021-12-06
Klassen-Diagramm Updaten	Wendelin Haller	100%	2021-12-09	2021-12-09
Zeitplan erstellen	Wendelin Haller	100%	2021-12-13	2021-12-13
AsteroidenWand-Klasse erstellt	Wendelin Haller	100%	2021-12-14	2021-12-14
Gegner und Movement erstellen.	Wendelin Haller	100%	2022-01-03	2022-01-03
Projektil und schiesen Methode	Wendelin Haller	100%	2022-01-03	2022-01-03
Level Ende Erstellen.	Wendelin Haller	100%	2022-01-03	2022-01-03
Wand und Colision erstellen	Wendelin Haller	100%	2022-01-03	2022-01-03
Interface erstellen	Wendelin Haller	100%	2022-01-04	2022-01-04
Classendiagramm Updaten.	Wendelin Haller	100%	2022-01-04	2022-01-04
Level 1/2 erstellen	Wendelin Haller	100%	2022-01-10	2022-01-10
Level 1/2 fertigstellen	Wendelin Haller	100%	2022-01-11	2022-01-11
Testen und Verbessern				
UnitTests und Mokup Tests erstell	Wendelin Haller	100%	2022-01-11	2022-01-11
Tests weiter machen	Wendelin Haller	100%	2022-01-11	2022-01-11
Tests fertig gemacht	Wendelin Haller	100%	2022-01-17	2022-01-17
Dokumentation				
JavaDoc angefangen	Wendelin Haller	100%	2022-01-18	2022-01-18
JavaDoc fertig gemacht	Wendelin Haller	100%	2022-01-20	2022-01-20
Dokumentation erstellen	Wendelin Haller	10%	2022-01-21	2022-01-25
Task 4	Wendelin Haller		date	date
Task 5	Wendelin Haller		date	date

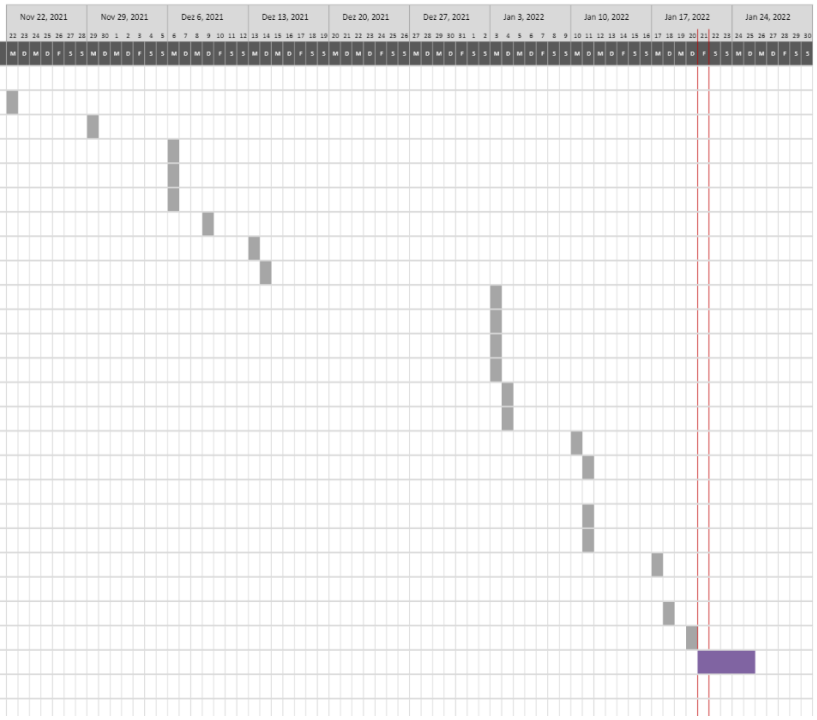


Abbildung 2 : Zeitplan 21.01.2021(siehe vergrösserte Version im Anhang.)

4. Entscheiden

Die folgende Abbildung 3 zeigt das Klassen-Diagramm meines Projektes.

Insgesamt sind es 13 Klassen und 2 Interfaces. Es gibt 8 Klassen, die von einer anderen Klasse in diesem Diagramm erbt.

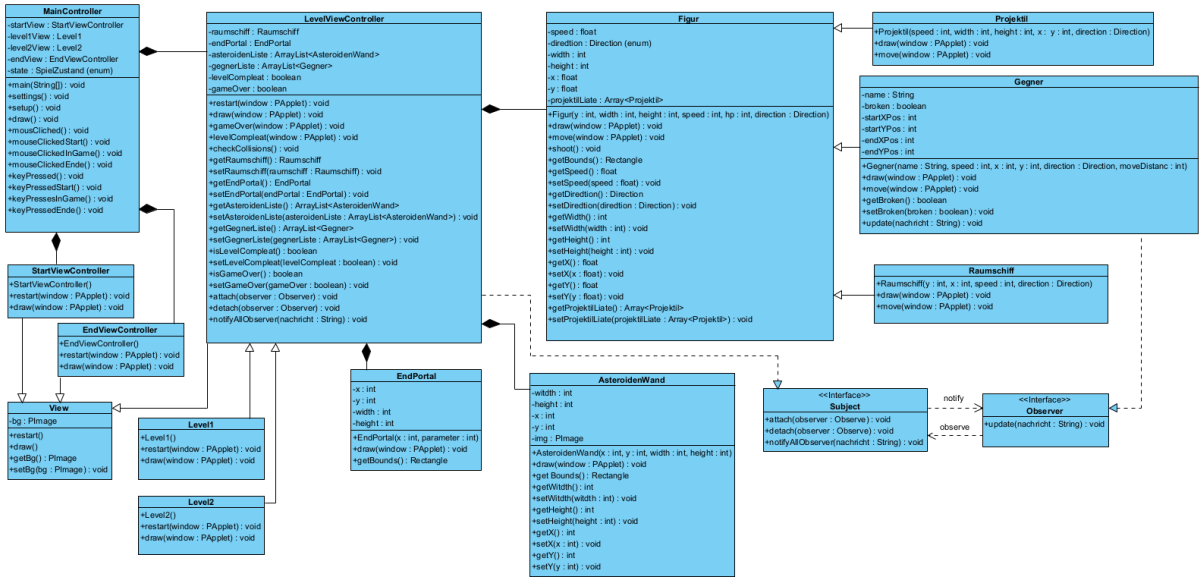


Abbildung 3 : Klassen-Diagramm

Die Observer-Interface wird von der Gegner-Klasse implementiert und die LevelView-Controller-Klasse implementiert das Subject-Interface. Die Interfaces sind dazu da, dass der LevelViewController-Klasse alle Objekte der Gegner-Klasse alarmieren kann, wenn einer der Gegner-Objekte von einem Projektil-Objekt getroffen wird, so dass dann die Gegner-Objekte reagieren können.

Von der abstrakten View-Klasse erben die StartViewController-Klasse, die EndView-Controller-Klasse und die LevelViewController-Klasse, weil alle diese drei Klassen haben, drei Sachen gemeinsahm:

- Das Attribut für das Hintergrundbild plus Getter und Setter.
- Die draw-Methode, die den Bildschirm zeichnet.
- Die reset-Methode ist dazu da, dass der Bildschirm gezeichnet werden kann, noch alle Objekte der AsteroidenWand-Klasse, der Gegner-Klasse, der EndPortal-Klasse und der Raumschiff-Klasse neu initialisiert werden.

Von der LevelViewController-Klasse erben die Level1-Klasse und die Level2-Klasse, weil die einzelnen Levels alle die gleichen Attribute und Methoden brauchen wie die LevelViewController-Klasse. Das Einzige, dass sich zwischen den Levels ändert sind die Implementierung der draw und reset-Methoden.

Die Klassen Gegner, Raumschiff und Projektil erben von der Figur-Klasse, weil sie bewegende Objekte sind und deswegen alle eine move-Methode, ein speed-Attribut und ein direction-Attribut brauchen.

5. REALISIEREN

Hier folgen Code-Snippets und eine Erklärung dazu. Die Code-Snippeds enthalten nicht immer JavaDoc, weil es sonst nur den Code zu lange macht.

5.1. VERERBUNG

Bei diesen zwei folgende Code-Snippeds sieht man gut wie die Raumschiff-Klasse von der Figur-Klasse erbt, weil bei der Raumschiff-Klasse das Stichwort «extends» und danach Figur steht und die Figur-Klasse das Stichwort «abstract» enthält, auch wenn nicht jede Parent-Klasse das Stichwort «abstract» enthält.

Ein Ausschnitt aus der Raumschiff-Klasse

```
public class Raumschiff extends Figur{
          public Raumschiff(int speed, int x, int y, Direction direction, PApplet window) {
                    super(speed, 50, 50, x, y, direction);
                    setImg_N(window.loadImage("/img/spaceship_N.png"));
                    setImg_E(window.loadImage("/img/spaceship_E.png"));
                    setImg_S(window.loadImage("/img/spaceship_S.png"));
                    setImg_W(window.loadImage("/img/spaceship_W.png"));
          }
```

Ein Ausschnitt aus der Figur-Klasse

```
public abstract class Figur {

          private int speed;
          private int width;
          private int height;
          private int x;
          private int y;
          private PImage img_N;
          private PImage img_E;
          private PImage img_S;
          private PImage img_W;

          private ArrayList<Projektil> projektilListe = new ArrayList<>();

          Direction direction = Direction.N;

          public enum Direction {
                    N,E,S,W;
          }

          public Figur(int speed, int width, int height, int x, int y, Direction direction) {
                    this.speed = speed;
                    this.width = width;
                    this.height = height;
                    this.x = x;
                    this.y = y;
                    this.direction = direction;
          }
```

5.2. ABSTRAKTE-KLASSE

An diesem Beispiel der View-Klasse sieht man gut, wofür ich diese Klasse abstrakt gemacht habe, weil diese Klasse überhaupt keine Funktionalität enthält und es sich nicht lohnt von solch einer Klasse ein Objekt anzulegen, weil die wichtigsten Methoden nicht in dieser Klasse implementiert werden sollen.

Die abstrakte View-Klasse

```
package controller;

import processing.core.PApplet;
import processing.core.PImage;

/**
 * Dieses abstracte Klasse ist für alle Views zuständig(Parent-Class).
 * @author Wendelin
 *
 */
public abstract class View {

	private PImage bg;

	/**
	 * Diese Methode soll nur einmahl vor der draw-Methode
	 * ausgefürt werden um alles zu reseten.
	 * @param window : PApplet
	 */
	public abstract void restart(PApplet window);

	/**
	 * Diese Methode zeichnet den View.
	 * @param window : PApplet
	 */
	public abstract void draw(PApplet window);

	/**
	 * @return the bg
	 */
	public PImage getBg() {
		return bg;
	}
	/**
	 * @param bg the bg to set
	 */
	public void setBg(PImage bg) {
		this.bg = bg;
	}
```

5.3. INTERFACE

Die folgenden zwei Interfaces kommunizieren miteinander indem sich der Observer dem Subject anschliesst mit der attach-Methode. Dann kann das Subject alle Observer kontaktieren mit der notifyAllObserver-Methode, falls es irgendeinen Grund gibt allen Observern eine Nachricht zu «senden».

Das Subjekt-Interface

```
package controller;

/**
 * Das ist ein Interface Subject.<br>
 * Dieses Interface arbeitet mit dem Observer-Interface zusammen.
 * @author Wendelin
 *
 */
public interface Subject {

	/**
	 * Diese Methode fügt dem Subject ein Observer-Object hinzu
	 * mit dem es späther kommunizieren kann.
	 * @param observer : Observer
	 */
	public void attach(Observer observer);

	/**
	 * Diese Methode entfernt dem Subject ein Observer-Object.
	 * @param observer : Observer
	 */
	public void detach(Observer observer);

	/**
	 * Diese Methode benachrichtigt alle Observer-Objecte
	 * mit denen die hinzugefügt wurden.
	 * @param nachricht : String zu übermitelnde Nachricht
	 */
	public void notifyAllObserver(String nachricht);

`
```

Das Observer-Interface

```
package controller;

/**
 * Das ist ein Interface Observer.<br>
 * Dieses Interface arbeitet mit dem Subject-Interface zusammen.
 * @author Wendelin
 *
 */
public interface Observer {

	/**
	 * Diese Methode updated den Observer je nach Nachricht die er bekommt.
	 * @param nachricht die übermitelt wird.
	 */
	abstract void update(String nachricht);

}
```

5.4. COLLISION

Im folgenden Code-Snipped ist die checkCollision-Methode, die in der LevelViewController-Klasse existiert. Diese Methode wird in den einzelnen Levels in der draw-Methode aufgerufen. In der Methode wird mit der intersect-Methode der Rectangle-Klasse überprüft, ob sich zwei Rectangle-Objekte aufeinander liegen. Je nachdem ob dann ein «true» oder «fals» zurück kommt werden gewisse Methoden aufgerufen, um entsprechende Funktionalitäten ins Rollen zu bringen.

```
/**
 * Diese Methode überprüft ob etwas colidiert und
 * wenn es dazu kommt werden entsprechende Attribute verändert oder Methoden aufgerufen.
 */
public void checkCollisions() {

		Rectangle rBounds = raumschiff.getBounds();
		Rectangle eBounds = endPortal.getBounds();
		int eX = (int) eBounds.getX() + 25;
		int eY = (int) eBounds.getY() + 25;

		if (rBounds.contains(eX, eY)) {
			setLevelCompleat(true);
		}

		for (AsteroidenWand a : asteroidenListe) {

			Rectangle aBounds = a.getBounds();

			if (rBounds.intersects(aBounds)) {
				setGameOver(true);
			}
		}

		for (Gegner g : gegnerListe) {

			Rectangle gBounds = g.getBounds();

			if(g.isBroken()) {
				gBounds = new Rectangle(-100, -100, 50, 50);
			}

			for (Projektil p : raumschiff.getProjektilListe()) {

				Rectangle pBounds = p.getBounds();

				if (gBounds.intersects(pBounds)) {
					g.setBroken(true);
					notifyAllObserver("Jemand ist KO gegangen.");
				}
			}

			for (Projektil p : g.getProjektilListe()) {

				Rectangle pBounds = p.getBounds();

				if (pBounds.intersects(rBounds)) {
					setGameOver(true);
				}
			}

			if (rBounds.intersects(gBounds)) {
				setGameOver(true);
			}
		}


}
```

6. KONTROLLIEREN

Ich habe bei drei Klassen Unit Tests durchgeführt, Dabei habe ich bei allen den Konstruktor und die move-Methode getestet.

Die Tests habe ich bei den Klassen «Gegner», «Raumschiff» und «Projektil» durchgeführt und auch Mockup verwendet, um die PApplet-Objekte zu simulieren.

Die Tests sind alle Positiv herausgekommen(siehe Abbildung 4-6) und es sind keine Fehler aufgetreten die ich beheben müsste.

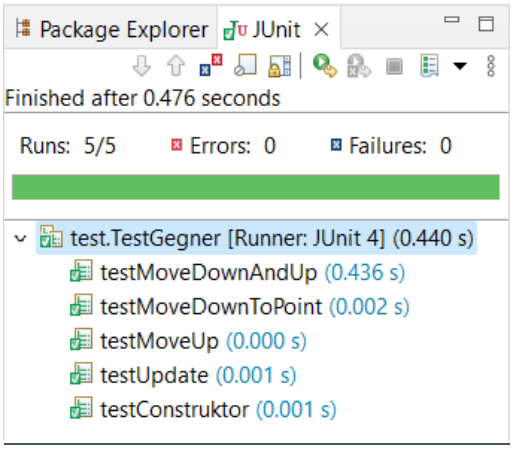


Abbildung 4 : TestGegner Resultat

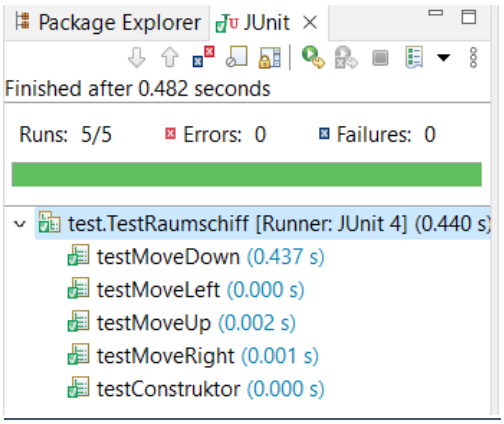


Abbildung 5 : TestRaumschiff Resultat

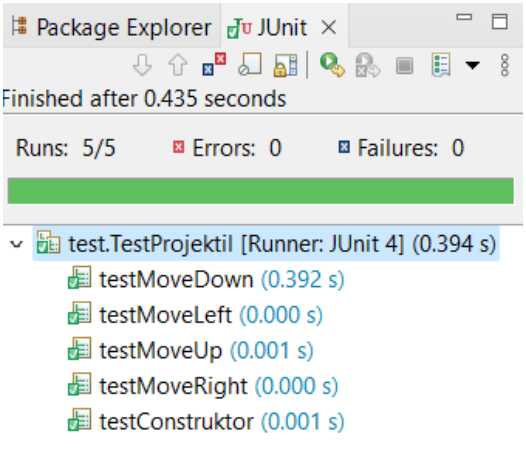


Abbildung 6 : TestProjektil Resulta

7. AUSWERTEN UND FAZIT

Bei diesem Projekt habe ich einiges gelernt und bin sehr mit dem Resultat zufrieden. Ich habe gelernt, wie man am besten mit einigen Fehlermeldungen umgeht und diese verhindern kann (java.util.ConcurrentModificationException).

Ich habe mich oft unterschätz, was den Zeitaufwand betrifft und möchte das bei einem nächsten Projekt besser machen, dass das Einplanen der einzelnen Aufgaben einfacher ist und ich nicht ungewiss bin wie viel Zeit ich noch aufwenden muss bis damit fertig bin.

Was habe ich gelernt?

- Wie Mann Markdown verwendet und was die Syntax ist für die wichtigsten Darstellungen wie Tabelle, Liste, Titel, Bilder oder Links ist.
- Ich weiss jetzt, wie man mit Git und GitHub arbeitet (alleine ohne Mittarbeiter).
- Wie man Status-Controll erstellen kann (mit einem Enum).
- Wie man Bilder für Spielfiguren einbindet (mit Processing).
- Wie man einen richtigen Zeitplan erstellt.
- Wie man Colision zweier Objekte überprüfen kann (mit der intersect-Methode der Rectangle-Klasse).
- Bilder nur einmal laden und nachher die Variable verwenden, weil sonst der Datenträger zu überlastet wird.

Es hat mir sehr viel Spass gemacht, dieses Projekt zu definieren und es dann zu Realisieren.

Anhang

Im Angang befinden sich Abbildungen, Tagesjournale und die eidesstattliche Erklärung. Die geschätzte Note und das Bewertungsraster ist den letzten Seiten zu entnehmen.

Die Quellenangaben sind dem Kapitel Informieren Seite vier zu entnehmen.

A. Abbildungen
B. Tagesjournal
C. Eidesstattliche Erklärung
D. Geschätzte Note und Bewertungsraster

A. Abbildungen

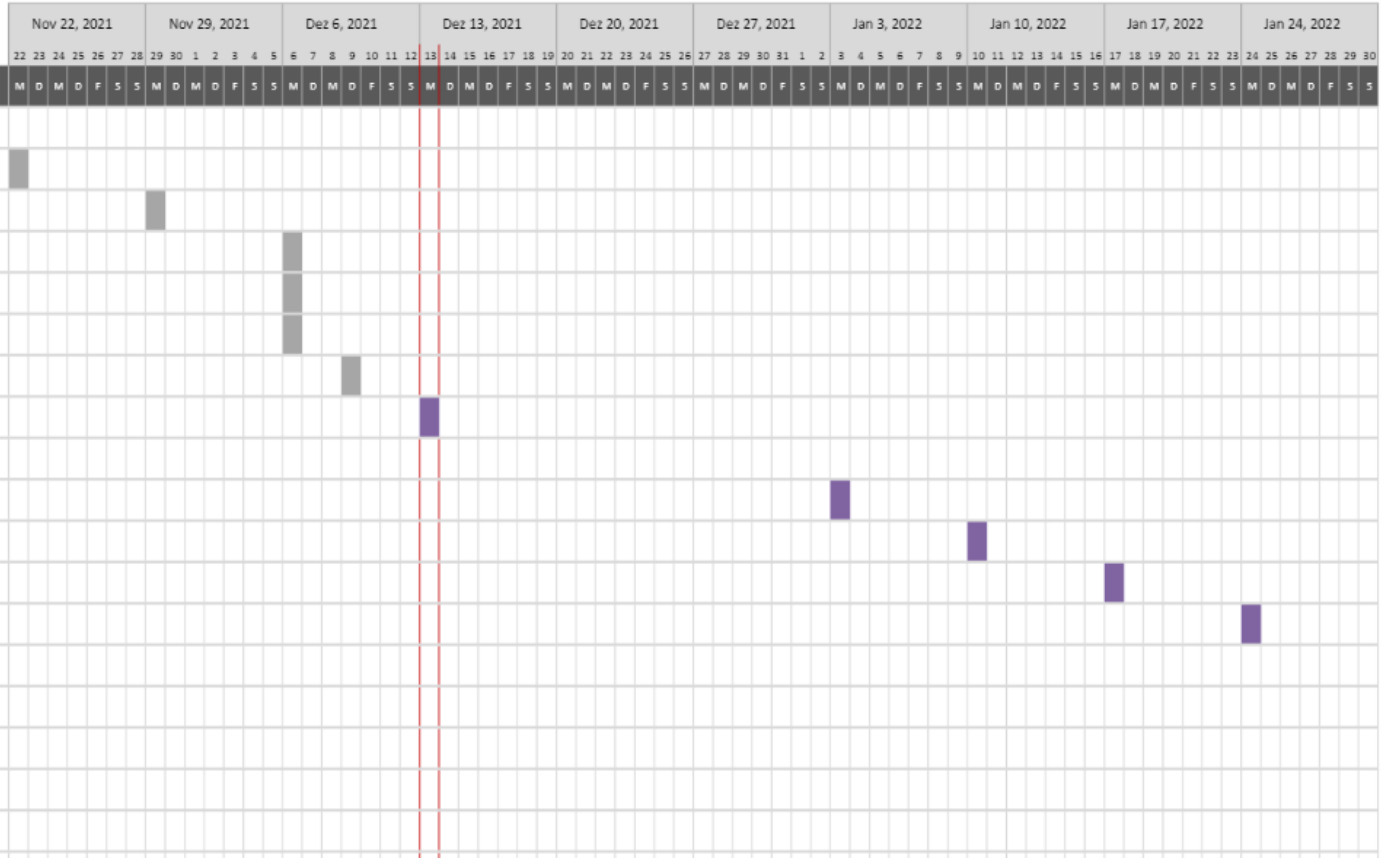
SpaceShip

Wendelin Haller

Project Start: Mo, 11.22.2021

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Start				
Projekt definiren	Wendelin Haller	100%	2021-11-22	2021-11-22
Aufbau erstellen	Wendelin Haller	100%	2021-11-29	2021-11-29
Klassen-Diagramm	Wendelin Haller	100%	2021-12-06	2021-12-06
Views erstellen	Wendelin Haller	100%	2021-12-06	2021-12-06
Raumschiff und Movement erstell	Wendelin Haller	100%	2021-12-06	2021-12-06
Klassen-Diagramm Updaten	Wendelin Haller	100%	2021-12-09	2021-12-09
Zeitplan erstellen	Wendelin Haller	50%	2021-12-13	2021-12-13
Implementieren				
Wand und Colision erstellen	Wendelin Haller	0%	2022-01-03	2022-01-03
Gegner und Movement erstellen.	Wendelin Haller	0%	2022-01-10	2022-01-10
Projektil und schiesen Methode	Wendelin Haller	0%	2022-01-17	2022-01-17
End eines Levels und EndView	Wendelin Haller	0%	2022-01-24	2022-01-24
Level 1/2 erstellen	Wendelin Haller	0%	2022-01-31	2022-01-31
Testen und Verbessern				
UnitTests und Mokup Tests erstell	Wendelin Haller		date	date
Fehler beheben	Wendelin Haller		date	date
Task 3	Wendelin Haller		date	date



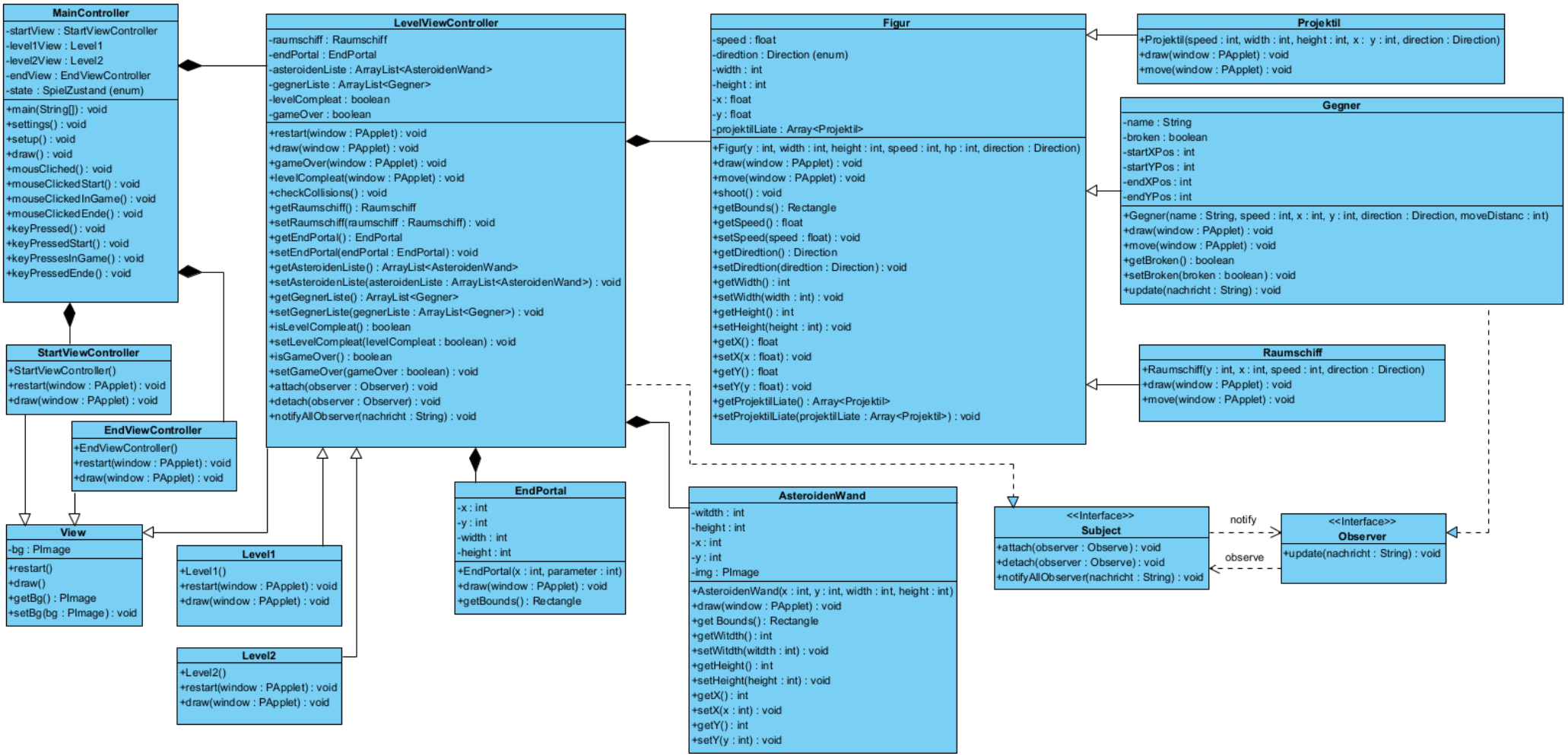
SpaceShip

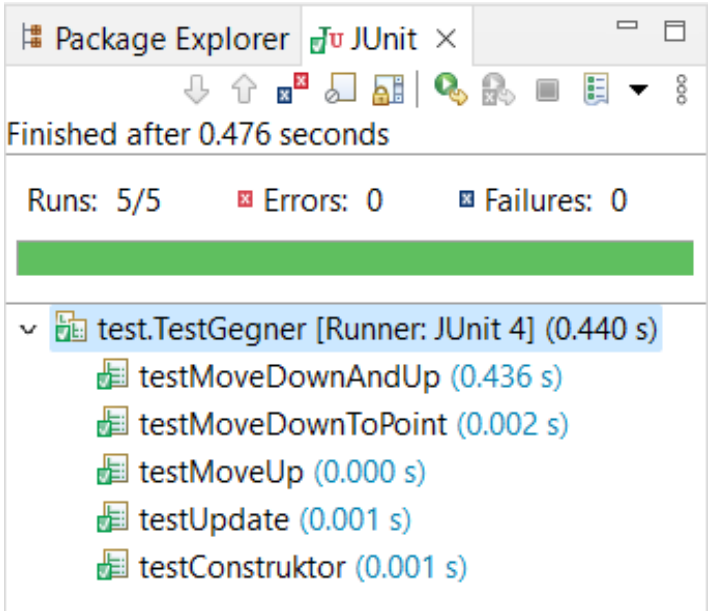
Wendelin Haller

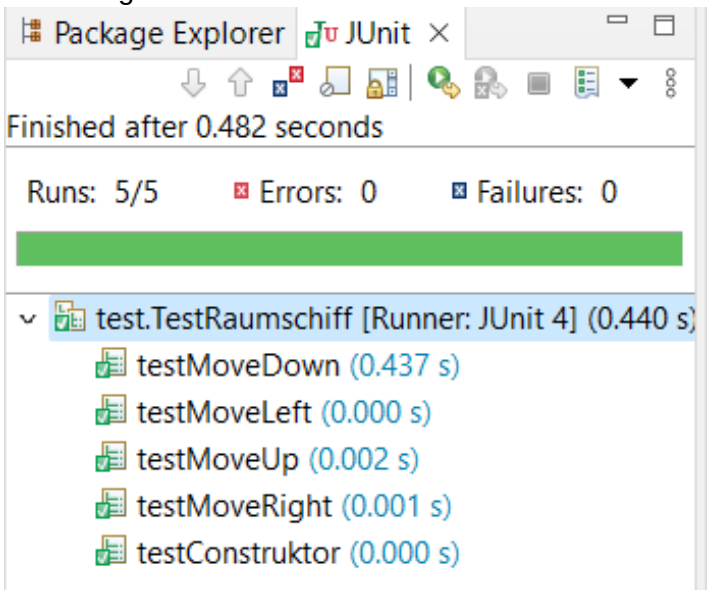
Project Start: Mo, 11.22.2021

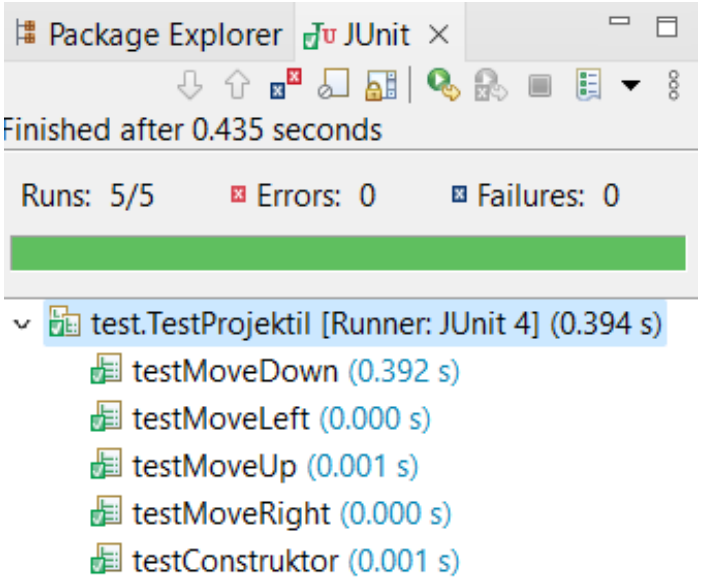
Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Start / Immplementieren				
Projekt definiren	Wendelin Haller	100%	2021-11-22	2021-11-22
Aufbau erstellen	Wendelin Haller	100%	2021-11-29	2021-11-29
Klassen-Diagramm	Wendelin Haller	100%	2021-12-06	2021-12-06
View's erstellen	Wendelin Haller	100%	2021-12-06	2021-12-06
Raumschiff und Movement erstell	Wendelin Haller	100%	2021-12-06	2021-12-06
Klassen-Diagramm Updaten	Wendelin Haller	100%	2021-12-09	2021-12-09
Zeitplan erstellen	Wendelin Haller	100%	2021-12-13	2021-12-13
AsteroidenWand-Klasse erstellt	Wendelin Haller	100%	2021-12-14	2021-12-14
Gegner und Movement erstellen.	Wendelin Haller	100%	2022-01-03	2022-01-03
Projektil und schiesen Methode	Wendelin Haller	100%	2022-01-03	2022-01-03
Level Ende Erstellen.	Wendelin Haller	100%	2022-01-03	2022-01-03
Wand und Colision erstellen	Wendelin Haller	100%	2022-01-03	2022-01-03
Interface erstellen	Wendelin Haller	100%	2022-01-04	2022-01-04
Classendiagramm Updaten.	Wendelin Haller	100%	2022-01-04	2022-01-04
Level 1/2 erstellen	Wendelin Haller	100%	2022-01-10	2022-01-10
Level 1/2 fertigstellen	Wendelin Haller	100%	2022-01-11	2022-01-11
Testen und Verbessern				
UnitTests und Mokup Tests erstell	Wendelin Haller	100%	2022-01-11	2022-01-11
Tests weiter machen	Wendelin Haller	100%	2022-01-11	2022-01-11
Tests fertig gemacht	Wendelin Haller	100%	2022-01-17	2022-01-17
Dokumentation				
JavaDoc angefangen	Wendelin Haller	100%	2022-01-18	2022-01-18
JavaDoc fertig gemacht	Wendelin Haller	100%	2022-01-20	2022-01-20
Dokumentation erstellen	Wendelin Haller	10%	2022-01-21	2022-01-25
Task 4	Wendelin Haller		date	date
Task 5	Wendelin Haller		date	date









B. Tagesjournal

C. Eidesstattliche Erklärung

Eidesstattliche Erklärung

Der Verfasser erklärt an Eides statt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als die angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

....

Ort, Datum Unterschrift des Verfassers

D. Geschätzte Note und Bewertungsraster

Wir wurden aufgefordert die Leistung zu schätzen.

Geschätzte Note: 5,5 bis 6

Das Bewertungsraster der Lehrperson folgt auf der nächsten Seite.

St. Gallen, 24. Januar 2022 Wendelin Haller