



Ein Minecraft Mod

in Java geschrieben, dank automatisierten Tools

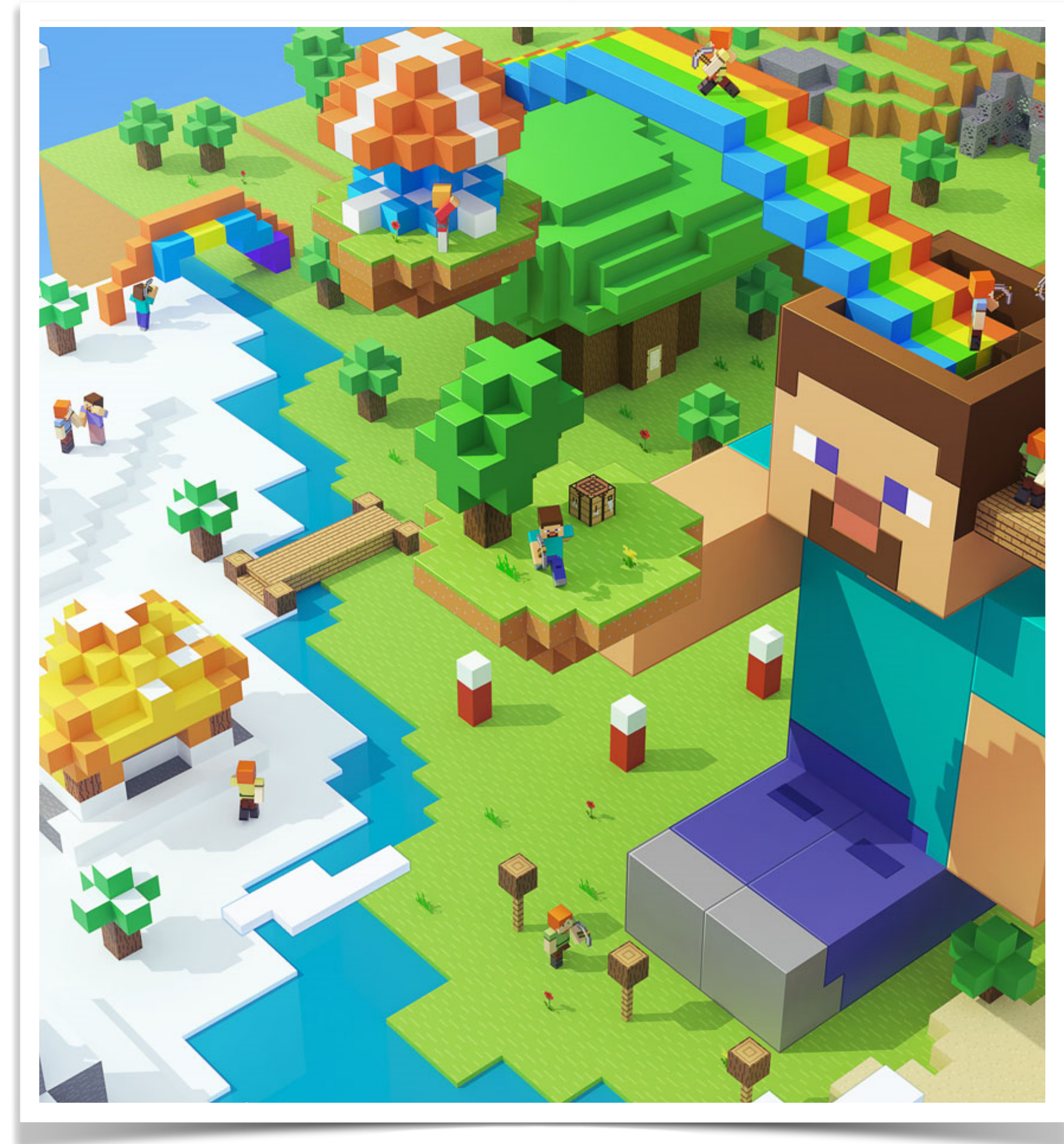
 <https://github.com/matthias-pmr/random-additions>

Automatisierte Software Entwicklung (SOSE2024)
Matthias Pommeray, Clara Pernet & Lucas Bodner (DFHI)
2024-07-22



Einleitung

- Ein Spiel, das wir gut kennen
 - die **Funktionsweise** des Spiels ist bereits bekannt
- Interesse am „Modding“
 - jeder kann Minecraft verbreiten
 - benutzt die **Tools**, die wir im Unterricht gesehen haben (Maven, Java...)
- Kann eine Herausforderung sein
 - eine **neue API** kennenlernen
 - braucht eine ganze Umgebung (**Server**)





Gliederung

1

Minecraft Modding

Arten und APIs

2

Gewählte Umgebung

Für die IDE und Minecraft

3

Projekt Beschreibung

Ideen von den Teilnehmern

4

Maven Konfiguration

Site, Warnings und Errors

5

Pre-Build

UnitTests und GH-Actions

6

Post-Build

In-Game Tests

★ Minecraft Modding

Unterschiedliche Arten

Mods

- auf dem **Client** & Server
- **neue Assets**, mehr Möglichkeiten
- dank **Gradle** 🐘 verwaltet



Bild: [Minecraft \(Youtube\)](#)

Plugins

- **NUR auf dem Server**
- benutzt, was schon existiert
- dank **Maven** 🦋 verwaltet



Bild: [Microsoft](#)



Frage an Sie

Wer hat schon Minecraft mit Modding gespielt?

(Oft laufen die Multiplayer Server mit Plugins, z.B. /trade)



★ Minecraft Modding

Plugin APIs und Serverssoftwares



[Link](#)

- Seit 2010
- Die erste weit verbreitete API
- CraftBukkit (das Serversoftware) wurde verlassen



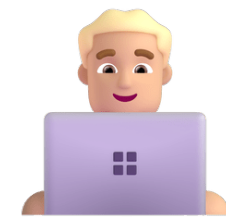
[Link](#)

- Seit 2012
- Immer beim weiterentwickeln der Bukkit API
- Heute noch sehr verbreitet



[Link](#)

- Seit 2017
- Legt auf Zuverlässigkeit & Leistung Wert
- Unterstützt noch die Spigot/Bukkit APIs, aber entwickelt seine eigene



Gewählte Umgebung

Für Minecraft (Version 1.20.6)

Die API



- Existiert seit lange (ist robust und viel Dokumentiert)
- Leicht verstehbar



- Existiert seit lange (alte Beschreibungen und vieles „deprecated“)
- Langsamer und begrenzt

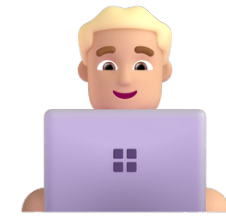


Die Serversoftware



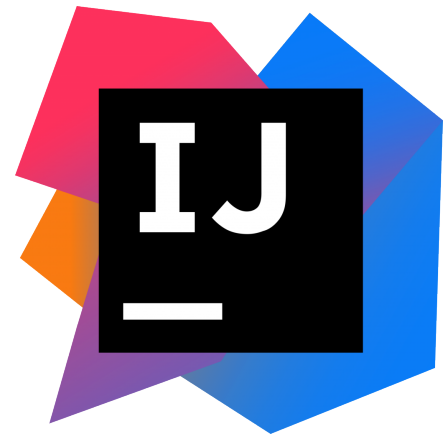
- Funktioniert für die letzte MC-Version
 - Modern und startet schnell
 - Unterstützt alle vorherigen APIs

- Bietet eine eigene API, die wir nicht benutzen



Gewählte Umgebung

Fürs Texteditor



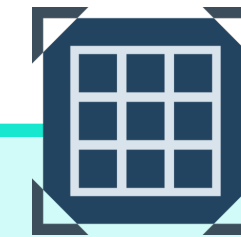
IntelliJ Idea

- zahlreiche Plugins
- bisher am meisten benutzt
- hat sehr nützliche Tools (Code → Generate etc.)



Code With Me

- erlaubt gleichzeitiges Coding
- daher verhindert Merge-Konflikts (bzw. der Bedarf an Branches) = bessere CI



Minecraft Development Plugin

- Gibt Basisimplementierungen für die unterschiedlichen APIs
- Verwaltet die Zielversionen (Paper, Bukkit, Minecraft)



Github Copilot

- Dank Kontext, hilft beim Autocomplete in repetitive Aufgaben
- Hallucinations = vorsichtig!



Error Prone Compiler

- Fügt einen neuen Compiler in IntelliJ Idea hinzu
- Identifiziert typische Fehlerlagen



google-java-format

- Formatiert den Code nach den Google-Standards
- Garantiert einen sauberer und lesbarer Code für alle

💡 Projektbeschreibung

Ideen

Matthias

- „Ein Kommando, das auf dem Chat ausgeführt wird“
- „Es gibt noch kein Taschenrechner in Minecraft!“



Lucas

- „Eine neue Möglichkeit, sich zu teleportieren, wenn der Spieler isst“
- „Eine Giftige Kartoffel zu essen, hat jetzt neue Folgen!“



Bild: minecraft.wiki

Clara



- „Dem Player etwas geben, wenn er ein Advancement fertig macht“
- „Das einfachste : Monster Hunter, und ein besseres Schwert geben (Diamond !)"



Gedanken

- Soll nicht zu schwierig werden = atomare Methoden, die leicht zu testen sind
- Features sollten von dem Spieler sichtbar sein



Frage an Sie

Wie macht Modding das Projekt besonders?



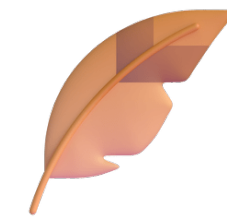
💡 Projektbeschreibung

Besonderheiten

- Keine `main()` Methode: `onEnable()` ist zuerst ausgeführt
- Viel mit „Events“ behandeln (Aktionen des Spielers erkennen)
- Keine sogenannte Anwendung, der Plugin ist auf einem Server zu laden
- Maven Shade benutzt, um die Paper (Bukkit) API einzubinden

```
/** This method is called when the plugin is enabled. */
@Override no usages 2 matthias-pmr +2
public void onEnable() {
    PluginManager pm = getServer().getPluginManager();
    pm.registerEvents(listener: this, plugin: this);
    pm.registerEvents(new FirstMonsterKilledReward(), plugin: this);
    pm.registerEvents(new PoisonousPotatoConsumptionRandomTeleport(), plugin: this);
    try {
        Objects.requireNonNull(this.getCommand(name: "calculator"))
            .setExecutor(new Calculator(this.getLogger()));
    } catch (NullPointerException e) {
        getLogger()
            .log(java.util.logging.Level.SEVERE, msg: "Could not register the command calculator", e);
    }
}
```





Maven Konfiguration

Warnings und Errors

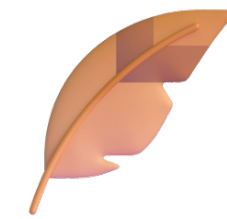
```
9      import org.bukkit.event.player.PlayerJoinEvent;  
10     import org.bukkit.plugin.PluginManager;  
11  
12     import org.bukkit.plugin.java.JavaPlugin;
```

```
[WARNING] src\main\java\eu\pommeray\randomadditions\RandomAdditions.java:[12,1] (imports) CustomImportOrder: S@paration suppl@mentaire dans le groupe d'importation avant 'org.buk  
kit.plugin.java.JavaPlugin'  
[WARNING] src\main\java\eu\pommeray\randomadditions\RandomAdditions.java:[24,7] (indentation) Indentation: Le fils de 'method def' au niveau d'indentation 6 n'est pas indent@ corr  
ectement, le niveau attendu est 4.  
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 3.309 s  
[INFO] Finished at: 2024-07-14T12:37:38+02:00  
[INFO] -----  
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-checkstyle-plugin:3.4.0:check (checkstyle) on project random_additions: You have 2 Checkstyle violations. -> [Help 1]  
org.apache.maven.lifecycle.LifecycleExecutionException: Failed to execute goal org.apache.maven.plugins:maven-checkstyle-plugin:3.4.0:check (checkstyle) on project random_additions  
: You have 2 Checkstyle violations.
```

```
23  @  public void onEnable() {  
24      PluginManager pm = getServer().getPluginManager();  
25      pm.registerEvents(listener: this, plugin: this);
```

Checkstyle plugin

→ Build Failure, wenn es Probleme mit dem Schreiben des Codes gibt.



Maven Konfiguration

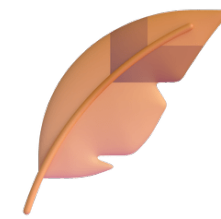
Warnings und Errors

Error Prone Compiler

→ Build Failure, wenn Probleme im Code,
wie ein `.toString()` auf ein Array

```
int[] coordinates = {x, z};  
String coordsString = coordinates.toString();  
// Adding + 1 to the y coordinate to prevent the player from being stuck in the ground  
int y = player.getWorld().getHighestBlockYAt(x, z) + 1;  
Location randomLocation = new Location(player.getWorld(), x, y, z);
```

```
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 8.511 s  
[INFO] Finished at: 2024-07-14T16:07:57+02:00  
[INFO] -----  
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.13.0:compile (default-compile) on project random_additions: Compilation failure  
[ERROR] /Users/lucasbodner/Documents/L2/Programmierung 1/random-additions/src/main/java/eu/pommeray/randomadditions/events/PoisonousPotatoConsumptionRandomTeleport.java:  
[ERROR] a:[30,29] [ArrayToString] Calling toString on an array does not provide useful information  
[ERROR] (see https://errorprone.info/bugpattern/ArrayToString)  
[ERROR] Did you mean 'String coordsString = Arrays.toString(coordinates);'?  
[ERROR]
```



Maven Konfiguration

Maven Site

random_additions
Last Published: 2024-07-12 | Version: 1.0-SNAPSHOT

Project Documentation
Project Information
Dependencies
Maven Coordinates
Distribution Management
About
Plugin Management
Plugins
Summary
Project Reports

Project Dependencies

compile

The following is a list of compile dependencies for this project. These dependencies are required to compile and run the application:

GroupId	ArtifactId	Version	Type	Licenses
com.github.seeseemelk	MockBukkit-v1.20	3.89.0	jar	MIT License
org.jacoco	jacoco-maven-plugin	0.8.12	jar	EPL-2.0
org.mockito	mockito-core	5.12.0	jar	MIT

test

The following is a list of test dependencies for this project. These dependencies are only required to compile and run unit tests for the application:

GroupId	ArtifactId	Version	Type	Licenses
org.junit.jupiter	junit-jupiter	5.10.2	jar	Eclipse Public License v2.0
org.mockito	mockito-junit-jupiter	5.12.0	jar	MIT

provided

The following is a list of provided dependencies for this project. These dependencies are required to compile the application, but should be provided by default when using the library:

GroupId	ArtifactId	Version	Type	Licenses
io.papermc.paper	paper-api	1.20.6-R0.1-SNAPSHOT	jar	-

RANDOM ADDITIONS
AUTOMATED DEVELOPED

Last Published: 2024-07-12 | Version: 1.0-SNAPSHOT

Project Documentation
Project Information
Dependencies
Maven Coordinates
Distribution Management
About
Plugin Management
Plugins
Summary

Project Dependencies

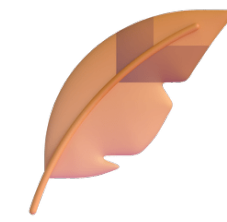
compile

The following is a list of compile dependencies for this project. These dependencies are required to compile and run the application:

GroupId	ArtifactId	Version	Type	Licenses
com.github.seeseemelk	MockBukkit-v1.20	3.89.0	jar	MIT License
org.jacoco	jacoco-maven-plugin	0.8.12	jar	EPL-2.0
org.mockito	mockito-core	5.12.0	jar	MIT

- Deployed und zu sehen auf:
<https://matthias-pmr.github.io/random-additions/>

- Expectations: Das Maven-Fluidio Skin anwenden
→ hat funktioniert, aber JaCoCo reports = unsichtbar
- Reality: sieht alt aus, aber funktioniert wie erwartet



Maven Konfiguration

Maven Site

Project Documentation

- Project Information
- Project Reports**
 - Surefire
 - JaCoCo
 - Checkstyle



Generated Reports

This document provides an overview of the va

Overview

Document
Surefire
JaCoCo
Checkstyle

Surefire Report					
Summary					
[Summary] [Package List] [Test Cases]					
Tests	Errors	Failures	Skipped	Success Rate	Time
9	0	0	0	100%	4.005 s

random_additions					
Element	Missed Instructions	Cov.	Missed Branches	Cov.	
eu.pommeray.randomadditions.commands		92%		80%	
eu.pommeray.randomadditions		88%		n/a	
eu.pommeray.randomadditions.events		100%		66%	
Total	18 of 337	94%	4 of 16	75%	

Checkstyle Results			
The following document contains the results of Checkstyle 10.17.0 with google_checks.xml ruleset.			
Summary			
Files	Info	Warnings	Errors
7	0	0	0

- Drei verschiedene Reports :
 - Jacoco fürs Coverage
 - Surefire für die Dauer des Tests
 - Checkstyle für die Konsistenz des Codes



Frage an Sie

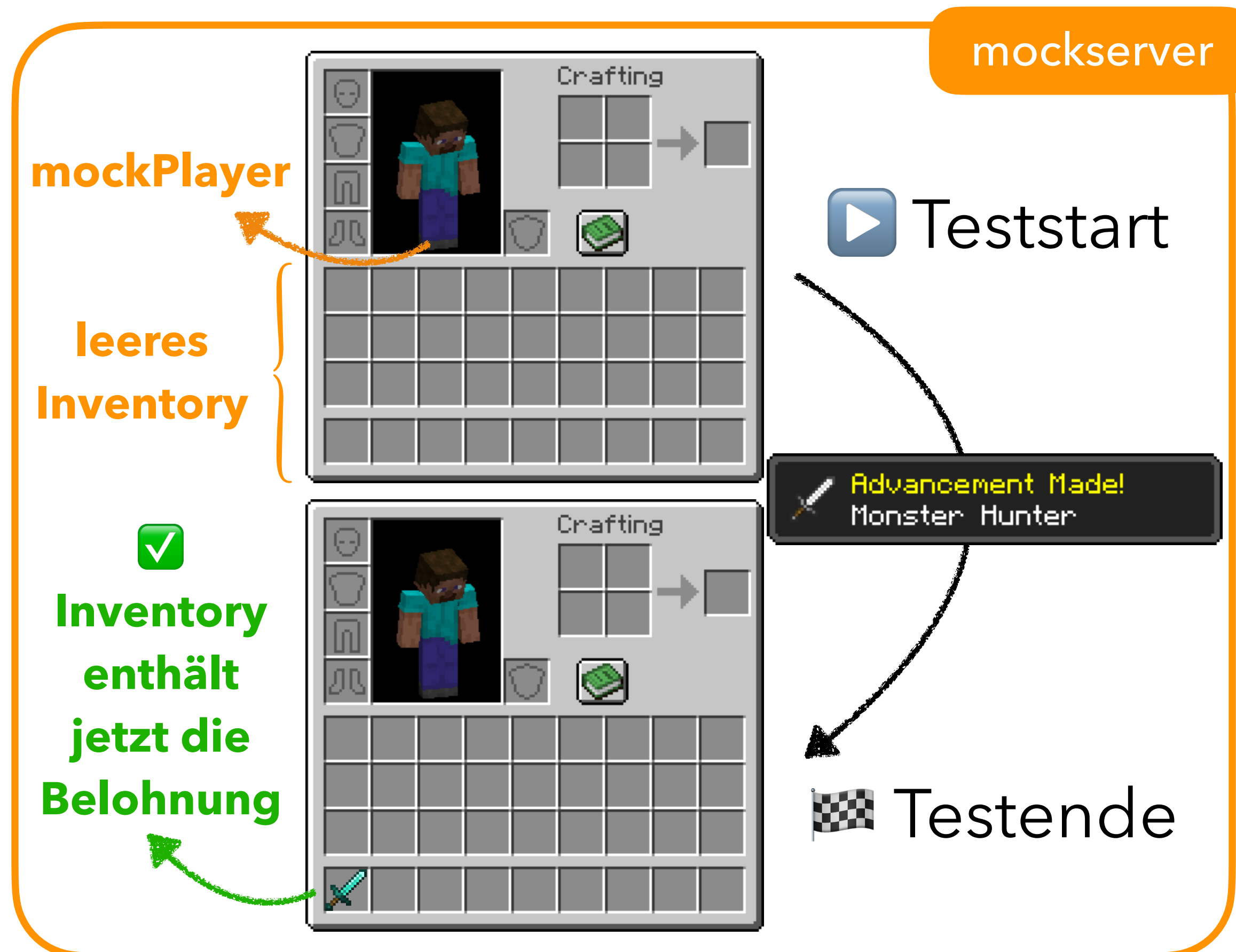
Wie testet man ein Spiel?





Pre-Build

UnitTesting



Benutztes Framework: **MockBukkit**

[Link](#)

- Schaffen von einer „Fake“ Umgebung: Server, Plugin, Players, Events...
- UnitTests ohne MockBukkit unmöglich!



Pain Points

- Schwieriges Einbinden von der Dependency (richtige Version finden + eine transitive Dependency ausschließen, die Probleme gebracht hat)
- Documentation weder vollständig noch hilfreich (viel „Trial and Error“)
- Was ist die Reihenfolge der Aufrufe? Was genau sollte „gefaked“ werden?

Pre-Build Github Actions

Build Project

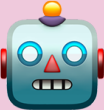
- Führt ein „`mvn -B package --file pom.xml`“ aus
- Schafft "artefacts" mit den Jars für das Server

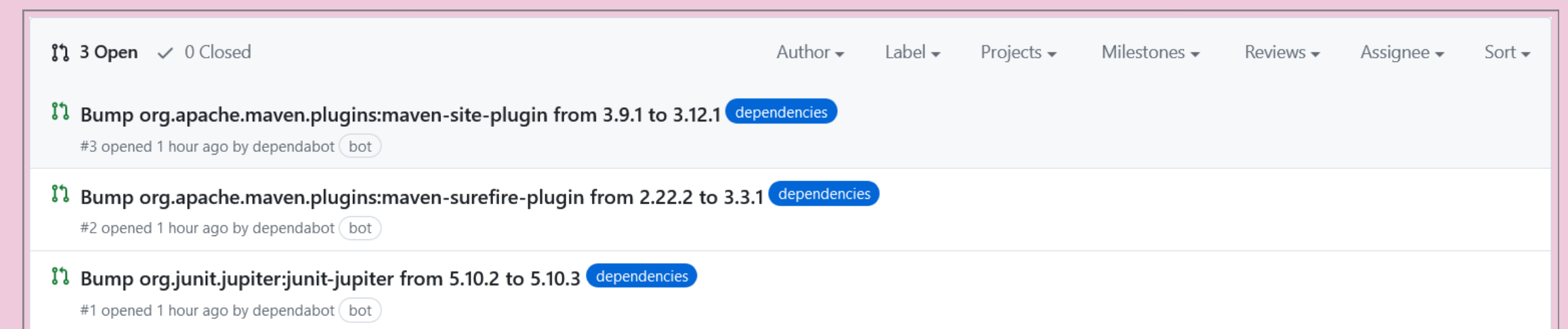
Unit Test

- Führt ein „`mvn -e --batch-mode --update-snapshots verify`“ aus
- braucht das „build“ Job

Deploy Maven Site

- Führt ein „`mvn clean verify jacoco:report site`“ aus
- Lädt die Website (`./target/site/`) auf die GH-Pages

-  **Dependabot:** tägliche Überprüfung
- „cherry picking“: manche Dependencies updates brechen das Projekt ab!



Post-Build

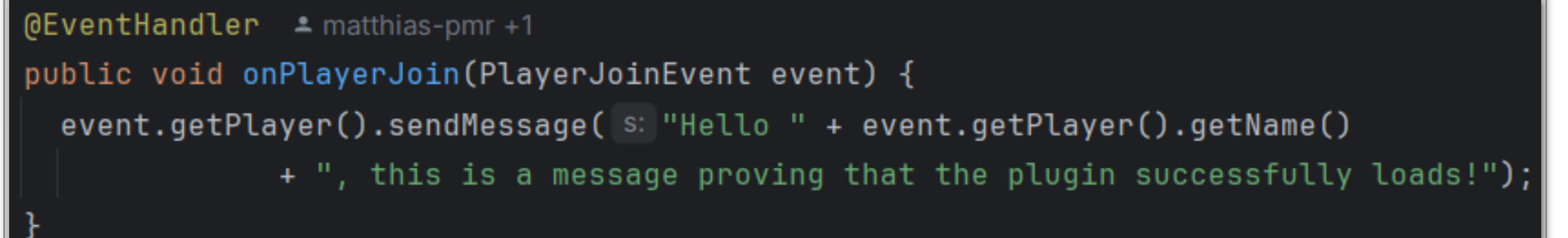
„In-Game“ Tests

- Nachricht in dem Chat, wenn der Player kommt
- Hilft um zu wissen, ob das Plugin läuft (oder nicht)
- nicht ein echter Test, sondern eine Hilfe



Hello Clarinette57, this is a message proving that the plugin successfully loaded!
Clarinette57 joined the game

automatische Nachricht
von Minecraft Servers



```
@EventHandler  @ matthias-pmr +1
public void onPlayerJoin(PlayerJoinEvent event) {
    event.getPlayer().sendMessage(s: "Hello " + event.getPlayer().getName()
        + ", this is a message proving that the plugin successfully loads!");
}
```


Post-Build

„In-Game“ Tests



- ✓ Die Hilfe ist richtig zum Spieler gezeigt, wenn er nicht genug Argumenten schreibt
- ✓ Alle Operationen sind richtig ausgegeben
- ✓ Die Farben sind richtig

Post-Build

„In-Game“ Tests

🤔 1000/-1000 ist extrem weit: der Server braucht Zeit, um der Player richtig zu „teleportieren“

✅ Man sieht, wie lange es dauert

✅ Man sieht die Distanz, und ob der Player IN oder AUF dem Boden teleportiert ist



Post-Build

„In-Game“ Tests



👉 Wenn mit dem
selben Player getestet:
„advancement revoke
<Player>
everything“ benutzt

✅ Man kann die
hinzugefügten Töne
(im Plugin) hören

✅ Die Farben sehen
gut aus

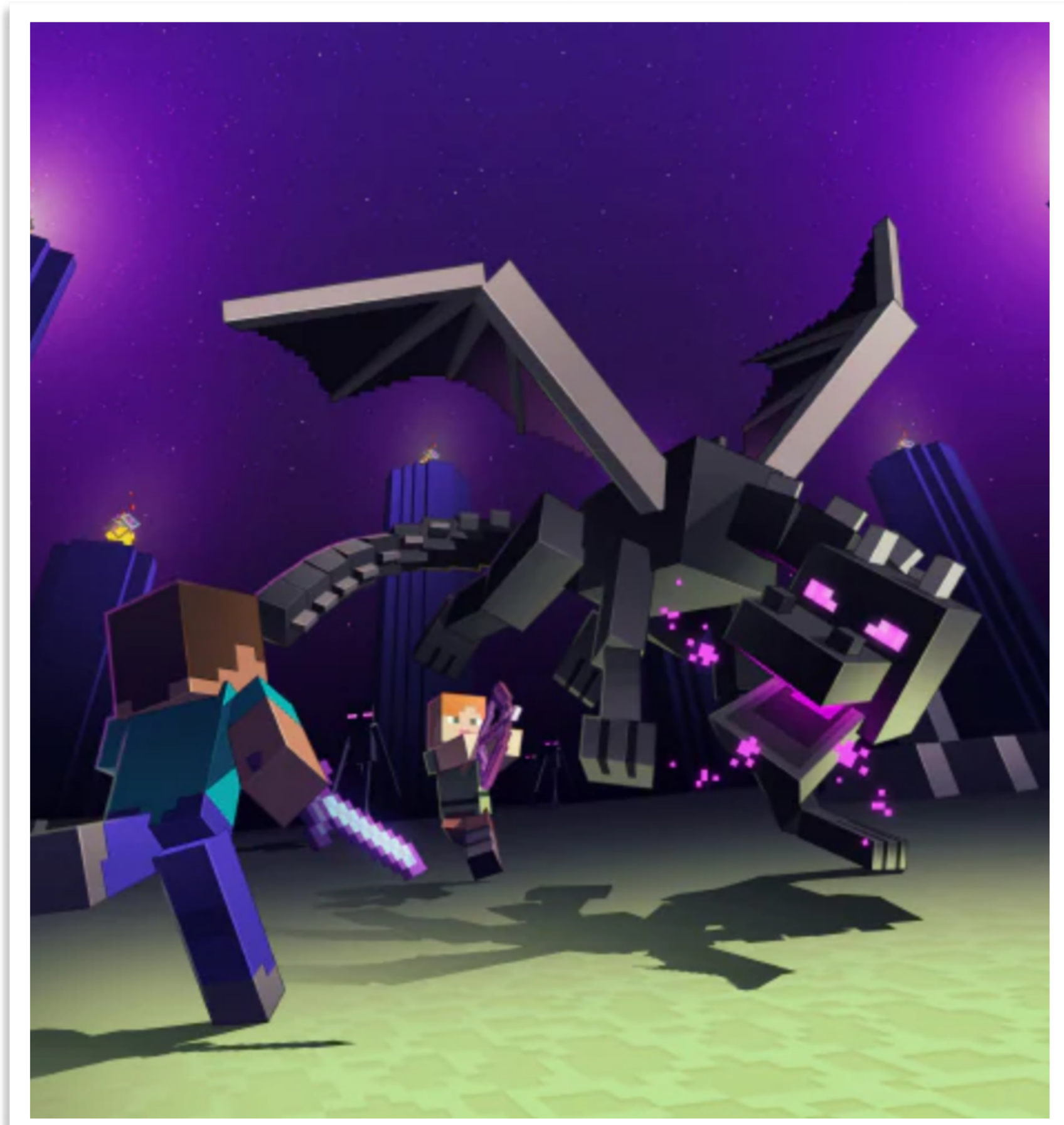


Bild: minecraft.wiki

Schluss

- Große Herausforderung
 - das Verhalten von Minecraft Plugins und neue **API kennenlernen**
 - das Jar nicht direkt ausführen, sondern **auf dem Spiel zu testen**
 - UnitTests für ein Spiel = **schwierig** ⚠
- Kohäsion
 - Benutzung von Github und **Code With Me**
 - **Nicht** immer miteinander **einverstanden**
 - Gute **CI**: Code With Me, GH-Actions, Artefacts, Maven Site
- Zukunft
 - viel über CI gelernt (**Merge Conflicts vermeiden...**)
 - wie man für ein Spiel programmiert (**Events**, Listeners...)



Frage an Sie

Verbesserungsvorschläge?

In der Umsetzung vom Projekt, die CI, die Tests...



Danke für eure Aufmerksamkeit
Haben Sie noch Fragen?