# ATMC-Projektplan: Erweiterung von End, Nether und Overworld

**(Projektlaufzeit: 6 Wochen)**

---

### Einleitung zur Minecraft-Mod "ATMC"

Die Minecraft-Mod **"ATMC"** (Advanced Terrain and Mob Customization) ist ein umfassendes Erweiterungsprojekt für die Fabric-Version 1.21.4, das darauf abzielt, die drei Hauptdimensionen von Minecraft – **End**, **Nether** und **Overworld** – mit neuen Inhalten, Mechaniken und Herausforderungen zu bereichern.

### Ziel der Mod

Unser Fokus liegt darauf, das Spielerlebnis in allen Dimensionen zu vertiefen und abwechslungsreicher zu gestalten. Während das **End** die Hauptaufmerksamkeit erhält und mit mystischen, außerirdischen Elementen glänzt, wird auch der **Nether** mit höllischen Landschaften und neuen Mobs erweitert. Schließlich erfährt die **Overworld** eine natürliche Evolution, die ihre Vielfalt und Tiefe steigert.

## Dimensionale Erweiterungen

**1. End (Primär)**

- **Thema:** Mystische und außerirdische Atmosphäre.
- **Inhalte:**
    - Kristalle, End-Erze, Pflanzen.
    - Schwebende Plattformen.
    - Neue Mobs.
    - Schwebende Ruinen, verlassene Festungen, Lost-City, Dungeons.
    - Ein Boss-Mob (z. B. End-Guardian).

**2. Nether (Sekundär)**

- **Thema:** Höllische, düstere Landschaften.
- **Inhalte:**
    - Magma-basierte Erze (Magma-Kristall), Lava-Interaktionsmechaniken.
    - Nether-Mobs (z. B. Nether-Guardians, Nether-Spiders, Nether-Bats).
    - Bastionsfragmente, Magma-Türme, Dungeons.

**3. Overworld (Tertiär)**

- **Thema:** Erweiterung der natürlichen Vielfalt.
- **Inhalte:**
    - Kristallhöhlen, verfluchte Wälder, magische Wälder.

- Saisonale Wettereffekte.
- Alte Tempel, Ruinen.
- Neue Mobs (z. B. Tiere, Waldgeister).

---

## Teamaufteilung und Verantwortlichkeiten

**1. Programmierung (1 Hauptperson, 3 Nebenpersonen)**

- **Verantwortlich für:** Mechaniken, Items, Blöcke, Rüstungen, Mobs, Strukturen, Generierung, etc...
- **Technologien:** Java (Fabric API).

**2. Modeling und Texturen (2 Personen)**

- **Verantwortlich für:** 3D-Modelle, Texturen, Animationen.
- **Tools:** Blockbench, Paint.NET.

**3. Strukturen und Weltgestaltung (1 Person)**

- **Verantwortlich für:** Design und Platzierung von Strukturen in den Dimensionen.
- **Tools:** Axiom, Structure Blocks.

---

**6-Wochen-Zeitplan**

### Woche 1: Grundlagen setzen

- **Programmierung:**
  - Mod-Struktur erstellen (ModID, Grundlogik).
  - Erste Items und Blöcke für das End (z. B. Kristalle, Pflanzen).
- **Modeling/Texturen:**
  - Texturen für End-Blöcke und Items erstellen.
  - Erste Basis-Mob-Modelle für das End beginnen.
- **Strukturen:**
  - Layouts für erste End-Strukturen (z. B. schwebende Ruinen).

---

### Woche 2: Erste Dimensionale Inhalte

- **Programmierung:**
  - Drop-Logik für End-Erze und Pflanzen.
  - Erste KI-Logik für einen End-Mob (z. B. Nether-Guardians).
- **Modeling/Texturen:**
  - Fertigstellung erster Mob-Modelle und Animationen.

    - Texturen für End-Items.
- **Strukturen:**
    - Grunddesign für die Lost-City im End.

---

### Woche 3: Fokus auf das End

- **Programmierung:**
    - Schwebende Plattformen.
    - Implementierung erster End-Mobs.
- **Modeling/Texturen:**
    - Fertigstellung von Texturen für End-Biom-Elemente.
    - Modellierung weiterer End-Mobs.
- **Strukturen:**
    - Integration von End-Strukturen (z. B. verlassene Festungen).

---

### Woche 4: Nether-Erweiterung

- **Programmierung:**
    - Lava-Interaktionsmechaniken (z. B. Magma-Mechaniken).
    - Zwei neue Nether-Mobs (z. B. Nether-Guardians, Nether-Spiders).
- **Modeling/Texturen:**
    - Texturen und Animationen für Nether-Mobs.
    - Rüstungs- und Waffendesign für Nether-Materialien.
- **Strukturen:**
    - Integration von Magma-Türmen und Bastionsfragmenten in den Nether.

---

### Woche 5: Overworld und Integration

- **Programmierung:**
    - Wetter- und Biomeffekte für die Overworld (z. B. Kristallhöhlen, verfluchte Wälder, magische Wälder).
    - Integration von Loot-Mechaniken und Crafting-Recipes/Smelting-Recipes.
- **Modeling/Texturen:**
    - Texturen für Overworld-Biomelemente erstellen.
    - Fertigstellung aller ausstehenden Mob-Modelle.
- **Strukturen:**

- Alte Tempel und Ruinen in die Overworld integrieren.

---

### Woche 6: Finalisierung und Tests

- **Programmierung:**
  - Balancing und Bugfixes.
- **Modeling/Texturen:**
  - Abschluss von Texturen, Animationen und finalen Details.
- **Strukturen:**
  - Letzte Anpassungen an Strukturplatzierung und Biomübergängen.

---

## Kommunikation und Tools

- **Tools:**
  - **Discord:** Für Meetings und regelmäßige Updates.
  - **GitHub/IntelliJ IDEA Community Edition:** Für Code- und Versionskontrolle.
  - **In-Game-Tools:** Axiom und Structure Blocks.
- **Meetings:**
  - Wöchentliche Fortschrittsbesprechungen (ca. 30 Minuten).

---

## Minecraft Modding erklärt

**Modding** in Minecraft bezeichnet das Erstellen und Bearbeiten von Spielinhalten, wie z.B. Blöcken, Items, Mechaniken oder ganzen Dimensionen, um das Spielerlebnis individuell anzupassen. Es ermöglicht Nutzern, ihre Kreativität auszuleben und das Spiel nach ihren Vorstellungen zu verändern.

### 1. Was ist Minecraft?

Minecraft ist ein beliebtes Sandbox-Spiel, bei dem die Spieler Blöcke abbauen und bauen können, um Strukturen, Landschaften oder ganze Welten zu erschaffen. Es gibt drei Hauptdimensionen: die **Overworld**, das **Nether** und das **End**. Das Spiel basiert auf einem einfachen Blockaufbau und nutzt eine anpassbare Spielwelt mit vielfältigen Mechaniken.

### 2. Was bedeutet Modding in Minecraft?

Modding (Kurzform von „Modifikation“) bedeutet, dass Spieler oder Entwickler das Spiel Minecraft um eigene Inhalte wie Blöcke, Items, Mechaniken, Ressourcenpakete oder komplette Erweiterungen erweitern. Mods können neue Features, Designs, Gameplay-Mechaniken oder auch Balance-Änderungen einführen.

### 3. Wie funktioniert Modding in Minecraft?

- **Modding-API**: In Minecraft werden Mods über APIs wie **Forge** oder **Fabric** eingebunden. Fabric ist eine Modding-API, die eine flexible und optimierte Unterstützung für Modder bietet.

- **Programmiersprache**: Die meisten Minecraft-Mods werden mit **Java** programmiert, da Minecraft in Java geschrieben wurde. Einige Mods nutzen spezielle APIs wie **Fabric API** oder **Forge** für erweiterte Mechaniken.
- **Werkzeuge**: Zum Modden verwendet man oft **Blockbench** für das Erstellen von 3D-Modellen, **Paint.NET** oder **Photoshop** für Texturen und **Axiom** oder **WorldEdit** für die Weltgestaltung.

## 4. Basis-Elemente beim Modding:

- **Blöcke und Items**: Mods können neue Blöcke und Items hinzufügen, die Spieler abbauen oder verwenden können.
- **Mechaniken**: Mods erweitern das Gameplay durch neue Funktionen, wie etwa neue Crafting-Rezepte, Angriffsmuster von Mobs oder besondere Wetterbedingungen.
- **Mobs**: Mods fügen neue Kreaturen oder feindliche Wesen (Mobs) hinzu, die Spieler in der Welt begegnen können.
- **Strukturen**: Mods können bestehende Strukturen erweitern oder neue Bauwerke in die Spielwelt integrieren.
- **Dimensionen**: Mods haben die Möglichkeit, neue Dimensionen wie z.B. das End oder Nether mit eigenen Biomen, Mobs und Strukturen hinzuzufügen.

## 5. Entwicklungsprozess einer Mod:

1. **Konzeption**: Entwickler überlegen, welche Inhalte sie in Minecraft modifizieren oder erweitern wollen – z.B. eine neue Dimension oder neue Blöcke.
2. **Entwicklung**: In der Entwicklungsphase programmiert das Mod-Team die gewünschten Mechaniken, baut 3D-Modelle, erstellt Texturen und strukturiert neue Spielinhalte.
3. **Testing**: Nach der ersten Fertigstellung erfolgt ein interner Test der Mod, um Fehler (Bugs) zu beheben und das Balancing zu überprüfen.
4. **Integration**: Die fertige Mod wird ins Spiel integriert, sodass Spieler die neuen Inhalte erleben können.
5. **Veröffentlichung**: Die Mod wird über Mod-Foren, Plattformen oder Webseiten für die Community freigegeben.

## 6. Warum Modding in Minecraft?

- **Kreativität fördern**: Modding erlaubt es Spielern und Entwicklern, ihre eigenen Ideen und Visionen direkt in das Spiel zu bringen.
- **Erweiterung der Spielwelt**: Mods erweitern die Spielwelt und sorgen für eine längere und tiefere Spielerfahrung.
- **Lernmöglichkeiten**: Beim Modding lernen Spieler technische und kreative Fähigkeiten wie Programmieren, Modellieren, Texturen erstellen und Leveldesign.

## 7. Fabric API:

Fabric ist eine moderne API, die eine flexible und benutzerfreundliche Modding-Plattform für Minecraft bietet. Sie ermöglicht es Moddern, in einem leistungsfähigen und schlanken Umfeld Mods zu entwickeln, ohne auf zusätzliche APIs oder Ressourcen angewiesen zu sein.

- Vorteile von Fabric:

- **Leichtgewicht**: Kleinere, schnellere und stabilere Mods im Vergleich zu anderen APIs.
- **Community**: Viel Unterstützung durch Tutorials, Guides und ein aktives Entwickler-Forum.
- **Flexibilität**: Erlaubt eine gute Anpassbarkeit und Modularität von Modifikationen.

## 8. Beispiele für Minecraft Mods:

- **OptiFine**: Eine Mod, die das Spiel mit verbesserten Grafikeinstellungen ausstattet.
- **JourneyMap**: Ein Mini-Map-Mod, der eine Karte der Spielwelt direkt im Spiel anzeigt.
- **Tinkers' Construct**: Mod, die das Crafting-System erweitert und neue Waffen und Werkzeuge ermöglicht.
- **Create**: Eine Mod, die das Spiel mit mechanischen Elementen erweitert, wie etwa Motoren und Maschinen.

## 9. Nutzung und Ethik beim Modding:

- **Eigene Kreationen**: Spieler und Modder können ihre Ideen eigenständig umsetzen und andere an ihrer Kreativität teilhaben lassen.
- **Community**: Mods fördern den Austausch zwischen Spielern und Entwicklern, wodurch neue Spielmechaniken oder Erlebnisse entstehen.
- **Fairness**: Die meisten Mods sind Open-Source oder kostenlos, was die Zusammenarbeit und den Austausch im Vordergrund stellt.

Mit dieser Erklärung möchten wir sicherstellen, dass Sie als Lehrer einen Überblick über den Modding-Prozess in Minecraft erhalten, um sowohl technisches Verständnis als auch kreative Arbeit in die Unterrichtseinheit einfließen zu lassen.

# SpionJam’s Arbeitsbereich:

In unserem Minecraft Mod Projekt ATMC übernehme ich die Verantwortung für den Großteil des Codes und setze die grundlegende Struktur der Mod auf. Das umfasst die Einrichtung der ModID, die Organisation der Projektdateien und die Sicherstellung, dass alle Abhängigkeiten und APIs korrekt eingebunden sind.

Im weiteren Verlauf konzentriere ich mich auf die Implementierung von zentralen Gameplay-Elementen wie:

- **Items:** Hinzufügen neuer Materialien, Werkzeuge und Gegenstände mit entsprechenden Rezepten und Eigenschaften.
- **Blöcke:** Erstellung von dekorativen und funktionalen Blöcken, einschließlich einzigartiger Verhaltensweisen wie interaktiven Eigenschaften oder besonderen Partikeleffekten.
- **Rüstungen:** Integration neuer Rüstungen mit individuellen Texturen, Werten und möglichen Spezialfähigkeiten.
- **Generierungen:** Implementierung von Weltgenerierungs-Features wie Erzvorkommen oder neuen Pflanzen, die speziell im End vorkommen sollen.

Zusätzlich werde ich mich auch um verschiedene **Texturen** kümmern, die für Items, Blöcke und Rüstungen benötigt werden. Diese Texturen müssen sorgfältig gestaltet und anschließend korrekt in den Code eingebunden werden.

Ein weiterer zentraler Aspekt meiner Arbeit ist die Erstellung und Pflege der **JSON-Dateien**, die für verschiedene Funktionen unerlässlich sind. Dazu gehören:

- **Item- und Block-Modelldateien:** Definition der visuellen Darstellung in der Spielwelt.
- **Rezepte:** Erstellung der Crafting- und Schmelzrezepte.
- **Loot-Tabellen:** Konfiguration, welche Gegenstände von Mobs oder in Kisten droppen.
- **Language-Dateien:** Sicherstellung der korrekten Lokalisierung aller neuen Inhalte, damit diese im Spiel mit Namen und Beschreibungen angezeigt werden.

Ich übernehme somit die technische Basis und die zentrale Implementierung der Mod, um sicherzustellen, dass die verschiedenen Inhalte nahtlos zusammenarbeiten und die Mod eine kohärente Erweiterung für Minecraft darstellt.

---

# Apocalypto_Torte’s Arbeitsbereich:

Ich bin ebenfalls für die Erstellung von **Strukturen** verantwortlich, aber mit einem stärkeren Fokus auf deren **Interaktion** mit anderen Objekten in der Spielwelt. Das bedeutet, ich programmiere, wo und wie neue **Strukturen** in der Welt erscheinen, sodass sie korrekt gefunden oder betreten werden können.

**Models und ihre Programmierung:**

Ein weiterer wichtiger Bereich ist die Erstellung von **Modellen**, die es aktuell im Spiel nicht gibt. Diese umfassen:

- Neue **Mobs** (z. B. Kreaturen oder Bossgegner).
- **Waffen** und andere Objekte.

Die von mir erstellten Modelle werden programmiert, sodass sie mit der **Spielmechanik** kompatibel sind.
**Beispiel:** Ein **mobiler Turm**, der als **Miniboss** agiert, könnte ein eigenes Modell mit speziellen Animationen und Angriffsmustern haben.

**Werkbänke und Rezepte:**

Ich kümmere mich auch um die Integration neuer **Werkbänke**. Dazu gehören:

- Das **Design** und die **Texturen** für die Werkbank.
- Programmierung der Werkbank, z. B. für **spezielle Funktionen** oder Animationen.
- Einbau von **Rezepten**, die auf der Werkbank genutzt werden können.

**Beispiel:**
Die **Unpure Crystallization Table** ist eine Werkbank, die als **Vorstufe** dient. Mit ihr kann man die **Amethyst-Kristall-Klinge** herstellen – ein **Schwert** mit einzigartigen Fähigkeiten.

- Um die **Pure Crystallization Table** herzustellen, muss der Spieler bestimmte **Items** finden. Diese zweite Werkbank hat andere, spezifische Eigenschaften.

---

# Jokestar’s Arbeitsbereich:

Ich konzentriere mich auf die Entwicklung von **Bossen** und **Mobs**, wobei ich sowohl deren **Modelle** als auch deren **Programmierlogik** erstelle. Jede Kreatur soll sich durch **einzigartige Eigenschaften** und **Verhaltensweisen** auszeichnen.

**Boss-Gegner:**

Boss-Gegner erhalten erweiterte **Fähigkeiten** und **Animationen**, um sie von Standardgegnern abzuheben.
**Beispiel:** Der **"Ender-Drache"** aus dem Basisspiel hat:

- **Angriffe** wie **Feuerbälle**, **Feuerspeien** und **Purge**.
- **Heilungskristalle**, die auf hohen Türmen stehen und zerstört werden müssen, um den Boss zu schwächen.

Ich werde ähnliche Mechaniken entwickeln, um Bosse **herausfordernd** und **abwechslungsreich** zu gestalten.

**Passive Mobs:**

**Passive Mobs** dienen als Quelle für **Nahrung** oder **Ressourcen**.
**Beispiel:**
Das **Ender-Schwein**:

- **Verhalten:** Es **teleportiert** sich weg, wenn es angegriffen wird, und wird dabei kurz **unsichtbar**.
- **Nutzen:** Es liefert **Fleisch** oder andere wertvolle **Ressourcen**.

**Neutrale Mobs:**

**Neutrale Kreaturen** greifen den Spieler nur an, wenn sie provoziert werden.
**Beispiel:**
Der **Pure Amethyst Golem**:

- **Verhalten:** Er bleibt **passiv**, bis er angegriffen wird. Danach greift er mit **starken Angriffen** an.
- **Nutzen:** Der Golem kann seltene **Ressourcen** fallen lassen, wenn er besiegt wird.

**Aggressive Mobs:**

**Aggressive Mobs** greifen den Spieler **sofort** an, sobald sie ihn entdecken. Diese Mobs sollen für die Spieler eine konstante **Herausforderung** darstellen und können in verschiedenen Bereichen auftauchen.
**Beispiel:**

1. **Ender-Spinne:**
    - **Verhalten:** Sie greift sofort mit **Sprungangriffen** an und kann sich kurzzeitig **unsichtbar** machen, um Spieler zu überraschen.
    - **Besonderheit:** Sie hinterlässt **Ender-Netze**, die den Spieler verlangsamen.
    - **Nutzen:** Ihre Überreste können für das Herstellen von **spezialisierten Bögen** oder **Werkzeugen** genutzt werden.
2. **Amethyst-Sentinel:**

- **Verhalten:** Ein mobiler Wächter, der auf Spieler zurast und sie mit einem **Schockangriff** lähmt.
- **Besonderheit:** Er ist in der Nähe von **Schreinen** und **Tempeln** zu finden und beschützt dort wertvolle **Schätze**.
- **Nutzen:** Nach dem Besiegen kann er **Kristall-Fragmente** droppen, die für mächtige **Crafting-Rezepte** verwendet werden können.

Zusätzlich kümmere ich mich darum, dass alle **Mobs** individuelle **Modelle** und **Animationen** erhalten. Der **Code** für ihre Bewegungen, Angriffe und Interaktionen wird ebenfalls von mir programmiert.

---

# Vykos’s Arbeitsbereich:

Meine Hauptaufgabe besteht darin, **Strukturen** für die Mod **vorzubauen** und diese anschließend zu **programmieren**. Dabei stelle ich sicher, dass die Strukturen im Spiel **korrekt generieren**. Das bedeutet:

- Die Strukturen erscheinen im **richtigen Biom**.
- Sie sind korrekt **orientiert**, damit sie beispielsweise nicht schief oder unnatürlich aussehen.
- Es werden die **passenden Blöcke** und Materialien verwendet, die zu ihrer Umgebung passen.

Neben den Strukturen kümmere ich mich auch um die Implementierung von **Waffen** und deren **Texturen**. Diese Waffen sollen **neue** oder **angepasste Funktionen** erhalten.
**Beispiel:** Ein **Dreizack**, der:

1. Einen **Strahl** abfeuert, der Gegnern **Schaden** zufügt.
2. Getroffene Gegner zum Spieler **heranzieht**, wodurch er sie leichter bekämpfen kann.

**Strukturen und Biome:**

Zu den Strukturen zählen auch größere Zusammenhänge wie **Biome**. Ein Biom hat spezifische Bedingungen, die programmiert werden müssen, z. B.:

- Welche **Blöcke** und **Materialien** dort erscheinen.
- Welche besonderen **Strukturen** das Biom enthält.
- Ob es spezifische **Pflanzen**, **Erze** oder andere Ressourcen gibt, die nur in diesem Biom vorkommen.

Ein **End-Dorf** oder eine **Ender-Burg** kann aus kleineren Komponenten bestehen, etwa:

- **Häusern**
- **Straßen**
- **Marktplätzen**
- **Türmen**

Diese Komponenten müssen zunächst **vorgebaut** werden. Anschließend programmiere ich, wie sie **zufällig**, aber **logisch sinnvoll** zusammengesetzt werden, damit sie **realistisch** aussehen. Das heißt:

- Häuser sind nicht ineinander verschachtelt.
- Straßen verbinden die Gebäude logisch miteinander.

**Beispiel für ein Biom:**

**Der Ender-Forest** ist ein einzigartiges Biom mit folgenden Eigenschaften:

1. **Bäume:** Neue Baumstrukturen, die z. B. aus **Ender-Holz** bestehen und auf ihren Ästen **Kristall-Früchte** tragen.
    - Diese Früchte können gesammelt und für das **Herstellen von neuen Items** verwendet werden.
2. **Strukturen:**
    - **Enderstädte**

- **Enderdörfer**
- **Türme** und **Schreine**
- **Tempel**

3. **Erze:** Das Biom enthält seltene Ressourcen wie **Ender-Erz**. Meine Aufgabe ist es, deren **Texturen** und die **Integration** in die Mod zu erstellen. Die Mechanik der Erze (wie Abbau oder Verarbeitung) wird jedoch von einem anderen Teammitglied übernommen.

Zusammenfassend bin ich dafür zuständig, **Strukturen** zu erstellen, **Texturen** für Waffen und Materialien zu bearbeiten und das Zusammenspiel von **Biomen** und **Strukturen** in der Spielwelt zu **programmieren**.

 <- ist eine Textur im Spiel, weshalb dies nicht viel Zeit brauchen würde.