[1 Ziel: Vom Piraten-Tileset zur Gameworld 1](#_Toc197461876)

[2 Glossar 2](#_Toc197461877)

[3 Übungsvorschlag 2](#_Toc197461878)

[3.1 Piratenwelt 3](#_Toc197461879)

[3.2 Ninja-Welt 3](#_Toc197461880)

[3.3 Sprout Welt 4](#_Toc197461881)

[4 Tileset- und Tilemap-Editor 4](#_Toc197461882)

[5 Wahrscheinlichkeiten (Gewichtungen) einzelner Tiles festlegen 5](#_Toc197461883)

[6 Ressourcen 6](#_Toc197461884)

# Ziel: Vom Piraten-Tileset zur Gameworld

**Ein Bild, das Text, Screenshot, Karte enthält.

KI-generierte Inhalte können fehlerhaft sein.**Ein Bild, das Schaltung, Elektronik, Elektronisches Bauteil, passives Bauelement enthält.

KI-generierte Inhalte können fehlerhaft sein.

Check Beispiel-Screenshots auf der Plattform ...

Ein Bild, das Text, Screenshot, Schaltung enthält.

KI-generierte Inhalte können fehlerhaft sein.

Es gibt auch ein RPG Tileset im gleich Stil:

<https://www.kenney.nl/assets/monochrome-rpg>

Ein Bild, das Muster, Zeichnung, Stoff enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Basiskonzepte

**TileMapLayer** hat 6 zu nutzende Eigenschaften

* tile\_set (PNG siehe oben)
* terrain\_set (Logische Gruppierung einzelner Tiles)
* z\_index (TileMapLayers verdecken sich u.U. was per z-Index korrigiert werden könnte)
* modulate (Verändert die Färbung des Layer, z.B. pinkes Schiff)
* collision\_enabled (tbd)
* navigation\_enabled (tbd)

**Terrainset**

Kontext für Übergänge (Anwendung der Bitmaskregeln), im vorliegenden Beispiel kommt dies jedoch nur begrenzt zur Geltung (z.B. das Ufer-Tiles zeigen automatisch den Übergang von Land zu Wasser). Methodisch würde ich zuerst eine große Wasserfläche definieren und dann die Landverteilung hinzufügen.

Beispielstruktur:  
Ein Bild, das Text, Schrift, weiß enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Atlas**

„Atlas“ in Godot (und allgemein in Game Development) stammt aus der Grafikprogrammierung und bezeichnet ein einzelnes Bild, das mehrere kleinere Grafikelemente (meist Tiles oder Sprites) enthält 🡺 quasi wie ein Karten Katalog

# Übungsvorschlag

**Tipp**

🡺 scharfe Pixelgrafik: Projekteinstellungen / Rendern / Texturen / Filter auf ‚Nearest‘ setzen (verhindert das Weichzeichnen beim Skalieren)

Empfohlene Projektstruktur (für ‚tileset Nutzung‘)

Ein Bild, das Text, Quittung, Schrift, weiß enthält.

KI-generierte Inhalte können fehlerhaft sein.

1. Baue eigene Pirat World
   1. Autotiling Ocean
   2. Autotiling Island
   3. Objekte / Schiff hinzufügen
   4. Animiere Kiste als Animated Sprite
2. Kopiere den ‘Animated Character’ vom letzten Seminar ins Spiel, dafür braucht es
   1. assets / character-sheet.png
   2. scenes / character.tscn + character.gd
   3. beide szenen ‘Character’ und ‘Pirate\_World’ in eine übergeordnete ‘Game World’ integrieren
   4. Game als ‘Hauptszene setzen’ & es sollte nur eine Camera mit Zoom geben
3. Verhindere das der Character über Wasser laufen kann (CollisionShapes)
   1. Ein Bild, das Text, Screenshot, Software, Multimedia enthält.

      KI-generierte Inhalte können fehlerhaft sein.Im entsprechenden ‘TileMapLayer’ das TileSet aufklappen und ein Physics Layer hinzufügen  
      (Kollision als physikalische Eigenschaft)
   2. Ein Bild, das Screenshot, Quadrat, Muster, Pixel enthält.

      KI-generierte Inhalte können fehlerhaft sein.Passe die CollisionShape der Palme an ...
4. .. versuche Dich in einer anderen Welt

Alle Welten sind im Github (\_asset\_packs) zu finden ...

## Piratenwelt

Startprojekt mit eigener Welt füllen ‘**03\_Pirates\_world\_begin’**

**Online**: <https://www.kenney.nl/assets/monochrome-pirates>

## Ninja-Welt

Github **03\_Ninja World**

Ninja / <https://pixel-boy.itch.io/ninja-adventure-asset-pack>

Ein Bild, das Clipart, Darstellung enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Sprout Welt

* <https://cupnooble.itch.io/sprout-lands-asset-pack>
* **03\_Sprout\_world**

Ein Bild, das Cartoon, Screenshot, Karte enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Tileset- und Tilemap-Editor

**Tileset Editor (Regeln für Tiles)**

* Zuweisung von Tiles zu bestimmten **Terrains** (z. B. „Gras“, „Weg“ definiert im TileMapLayer)
* Autotiling über Bitmasken definieren
* Wahrscheinlichkeiten definieren
* Collisionsbereiche definieren

Final können alle Einstellungen als .tres im Inspektor gespeichert werden und somit kann das Tilset unabhängig vom Knoten verwendet werden (d.h. der TileMapLayer lädt im Tileset die entsprechende .tres Datei) und beide Terrains stehen instant zur Verfügung und können im Tilemap-Editor verwendet werden. Sicherheitshalber sollten .png und .tres im selben Pfad stehen, z.B.:

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Tilemap Editor (Tiles platzieren)**

* Platziert Tiles per Pinsel, Rechteck- und Füllwerkzeug
* 2 Modi:
  + Terrains: Nutzen der Autotiling-Logik
  + Tiles: Einzelne Objektes aus dem Tilesheet platzieren

# Wahrscheinlichkeiten (Gewichtungen) einzelner Tiles festlegen

Alle Ocean Tiles haben dieselbe Gewichtung von 1,0 (Standardeinstellung):

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Angepasste Gewichtung:

Ein Bild, das Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Wahrscheinlichkeiten werden unter Tileset / Malen / Wahrscheinlichkeiten eingestellt:

* Je höher der Wert, desto häufiger wird das Tile angezeigt

Ein Bild, das Text, Screenshot, Multimedia-Software, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Ressourcen

Autotiling   
<https://docs.godotengine.org/en/4.4/tutorials/2d/using_tilesets.html>

Ohne Autotiling aber mit Collisionshapes

<https://www.youtube.com/watch?v=43sJIWaj2Yw&ab_channel=CocoCode>