# Backup Ressourcen

Offizielle Tutorials:

<https://docs.godotengine.org/de/4.x/about/introduction.html>

Godot Skript Referenz: <https://docs.godotengine.org/de/4.x/tutorials/scripting/gdscript/gdscript_basics.html>

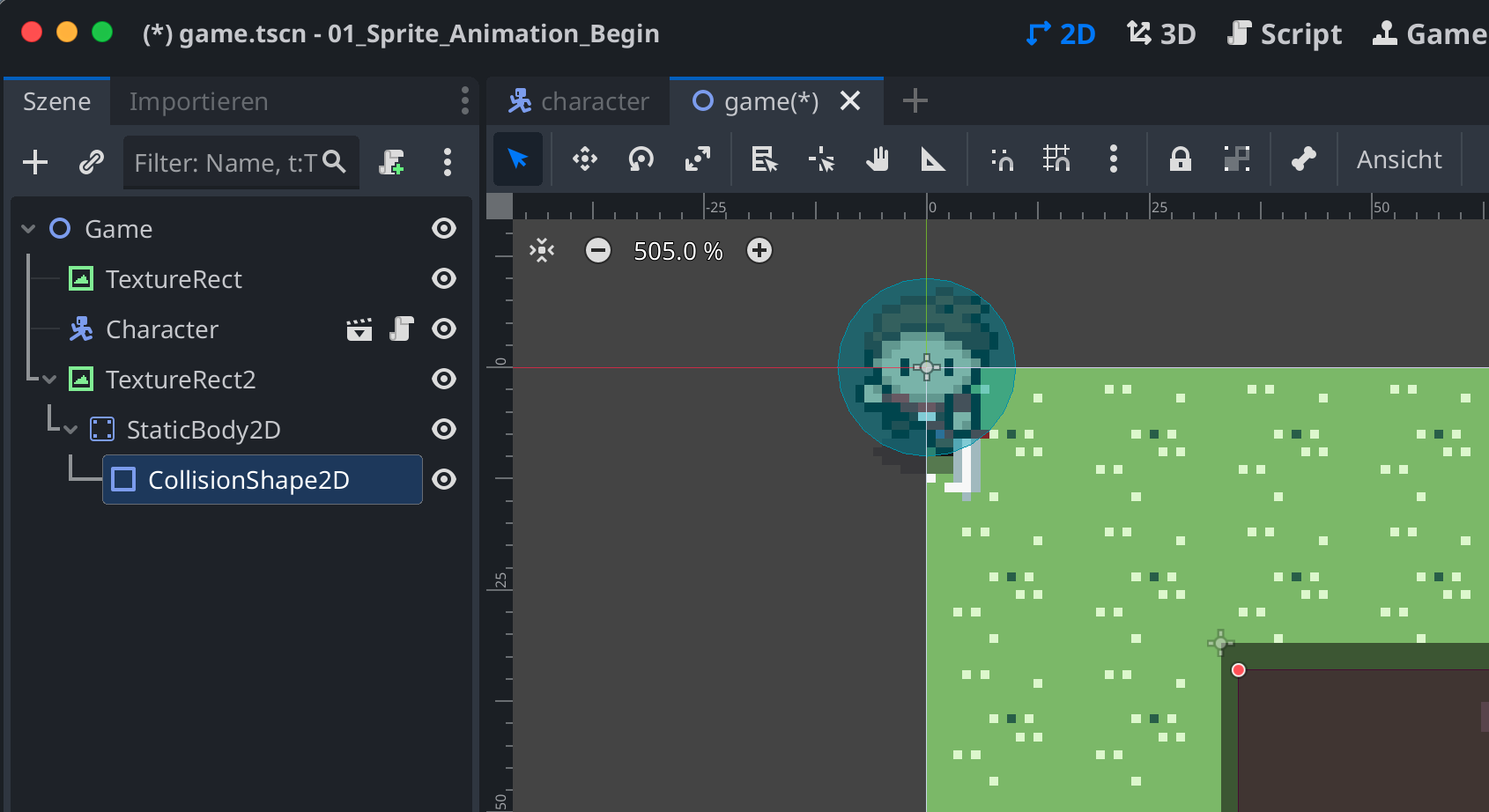
Überschaubares YouTube Tutorial (nicht exakt der im Folgenden beschriebene Ansatz)   
<https://www.youtube.com/watch?v=kw1iI69kW6o&ab_channel=CodeWithRo>

# Übungsziel

Teil 1: Ein Charakter läuft per Tastatursteuerung über einen Hintergrund, per *move\_and\_slide ()*

Teil 2: Ein Charakter läuft per Tastatursteuerung inklusive Laufanimation. Bestimmten Bereichen können nicht betreten werden (Charakter kann nicht über Bäume und Häuser laufen).

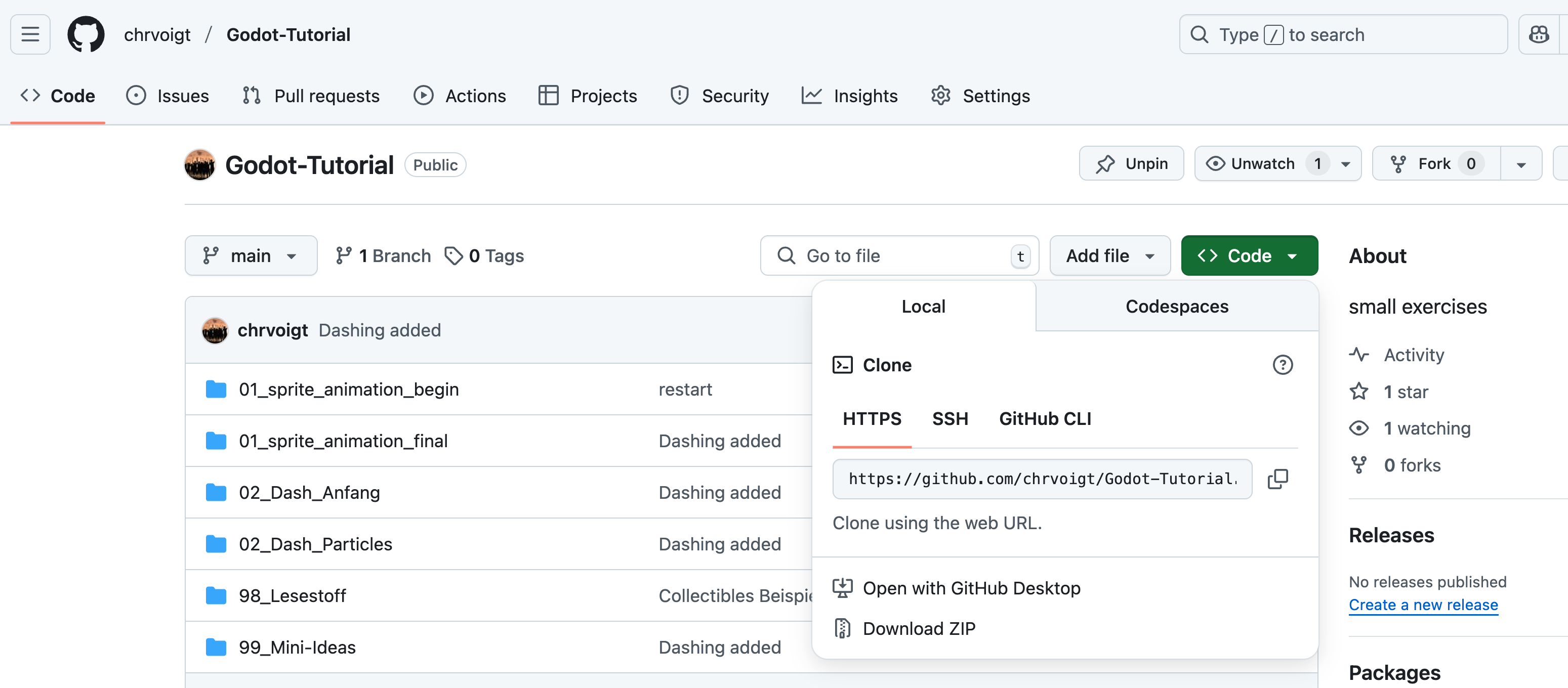
Im Überblick sieht das Endresultat dann so aus:



# Projekt starten

🡺 Ich würde empfehlen, nach dem runterzuladen und entpacken des Github Codes, das jeweilige Projekt **nochmal in einen Arbeitsordner zu kopieren** und dort zu arbeiten (auf diese Weise)   
Da es beim entpacken der Datei immer wieder mal zu Missverständnissen kommt, hier ein kurzer Praxistipp: <https://praxistipps.focus.de/zip-datei-entpacken-so-gehts_136887>

🡺 der link zum Github [https://github.com/chrvoigt/Godot-Tutorial#](https://github.com/chrvoigt/Godot-Tutorial) und hinter dem grünen Button verbirgt sich dann auch der ZIP Download



🡺 über den Github Code müsst ihr nur das jeweilige .projekt File mit Godot öffnen

🡺 Für eigen Projekte: Projektname generiert automatisch ein Unterverzeichnis

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

KI-generierte Inhalte können fehlerhaft sein.

# Character Grafiken bereitstellen

* Teil 1: Im ‚asset‘ Ordner, siehe ‚single\_character.gif‘
* Teil 2: In Aseprite oder Libresprite können Frames aus Bilddateien ausgeschnitten und angeordnet werden (sogenannte Spritesheets), jedes Sprite kann dann per Position (x, y) aufgerufen werden   
  (links oben = 0, 0 und rechts unten 3, 3) ... siehe ‚ character-sheet.png‘. Die Bewegung nach Links würde im Beispiel in der 4. Reihe bei (3,0) starten ...

Ein Bild, das Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Cartoon, Clipart, Animierter Cartoon enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Grundeinstellungen und Low Code Funktionalität

**Folgende Grundeinstellungen** werden häufig in der Projektumgebung vorgenommen (und sind im Beginner Pack ‚ 01\_sprite\_animation\_begin‘ schon voreingestellt:

1. Hauptszene und Projektname festlegen:   
   ‚Projekt / Projekteinstellungen / Anwendung / Ausführen / Hauptszene‘
2. Fensterdimensionen ‚Projekt / Projekteinstellungen / Anzeige / Fenster / Größe‘  
   x: 1200; y:800 (für neuere Rechner mit hoher Bildschirmauflösung können diese Werte auch verdoppelt werden)
3. Rendering für Pixel-Grafik einstellen, unter ‚Projekt / Projekteinstellungen / Rendern / Texturen‘  
   Default\_Texture\_Filter auf ‚nearest‘

In Abschnitt 6 wird noch die projektweite ‚Gravity‘ besprochen, die gilt nur für Side-Scroller und muss daher auf Null gesetzt werden.

**3 Hauptszenen werden angelegt**:

* ‚Character‘ erstellen (CharacterBody2D)
* ‚Games‘ erstellen (Node2D)
* Im Teil 2: kommt dann noch ‚World‘ (Node2D) hinzu

**Character Szene** bekommt 4 Nodes (siehe Image):

* Z.B. Sprite2D enthält das‚single\_character.gif‘
* Camera Zoom auf 3

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Game Szene** kombiniert**:**

* Hintergrund (TextureRect) mit dem entsprechenden PNG (green\_tile.png) aus dem ‚assets‘ Ordner verknüpfen und ‚**Stretch Mode**‘ auf ‚Tile‘ setzen

🡺 Reihenfolge ist entscheidend

Zukunftswissen: Eine komplexere **Character Szene** könnte auch so aussehen ...

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Script hinzufügen (unter Verwendung des Templates)

Um das Template einsetzen zu können, braucht es 2 Bedingungen

1. In dem jeweiligen Ordner darf es noch keine Datei mit dem entsprechenden Namen geben, z.b. ‚characetr.gd‘
2. Das Script muss zu einem CharacterBody2D Node hinzugefügt werden und ‚Vorlage‘ bekommt ein blaues Häkchen

🡺 danach sollte der Sprite sich zumindest auf der x-Achse bewegen ...

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

KI-generierte Inhalte können fehlerhaft sein.

## Scripte und ChatGPT

ChatGPT kann gut füe kleiner Code Snippets verwendet werden, dabei schelcihen sich aber leicht minimale Fehler ein. Hier ein Beispiel für einen ‚Flüchtigkeitsfehler‘, velocity.x sollet velocity.y sein:

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

KI-generierte Inhalte können fehlerhaft sein.

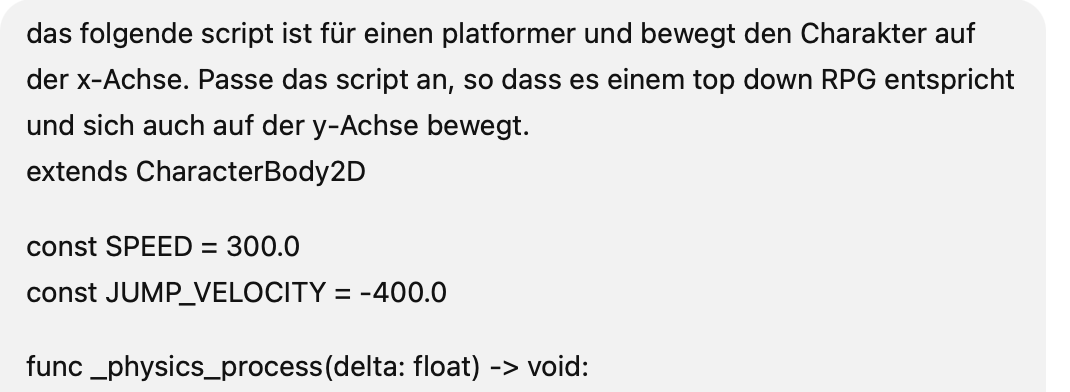
Beispiel für Code Optimierung:

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

Beispiel für Code-Generierung (Umwandlung des Templates):

🡺 erfordert oft noch ein minimales Nachjustieren, was ein einfaches ‚copy und paste‘ verhindert



## Manuelle Anpassung

Natürlich kann der Code auch manuell angepasst werden ...

Tipp: Speed = 300

* Alles was nach Plattformer aussieht auskommentieren #
* Die bestehende rechts – links Bewegung (direction\_x) erweitern auf hoch-runter erweitern (direction\_y)
* Scripte sind Erweiterungen bestehender Funktionalitäten, im vorliegenden Beispiel gibt es schon ein Funktion ‚*move\_and\_slide ()‘,* die wir über die Koordinaten *direction.x* und *direction.y* ansteuern können

Optional: Beispiel für einen ‚randomly moving‘ NPC

Quelle: <https://youtu.be/LMSbPkNgnWA?t=904> (ersten 13 min)

# Animation

* 5 animationen einrichten
  + Down, up, right, left, reset
* Jede Animation wird mit den gleichen Schritten bearbeitet
  + Parameter einrichten
    - Framedauer (nearest FPS): 0.1
    - Animationdauer: 0.4
    - Frames hinzufügen (nachdem das Spritesheet selektiert wurde)
    - Dauerschleife

# Animationen im Script integrieren

Zuerst wird im Script aus Kapitel 6.1 folgende Zeile eingebaut

*@onready var animation\_player: AnimationPlayer = $AnimationPlayer*

Und die dazugehörige \_physics\_process Funktion sieht z.B. so aus

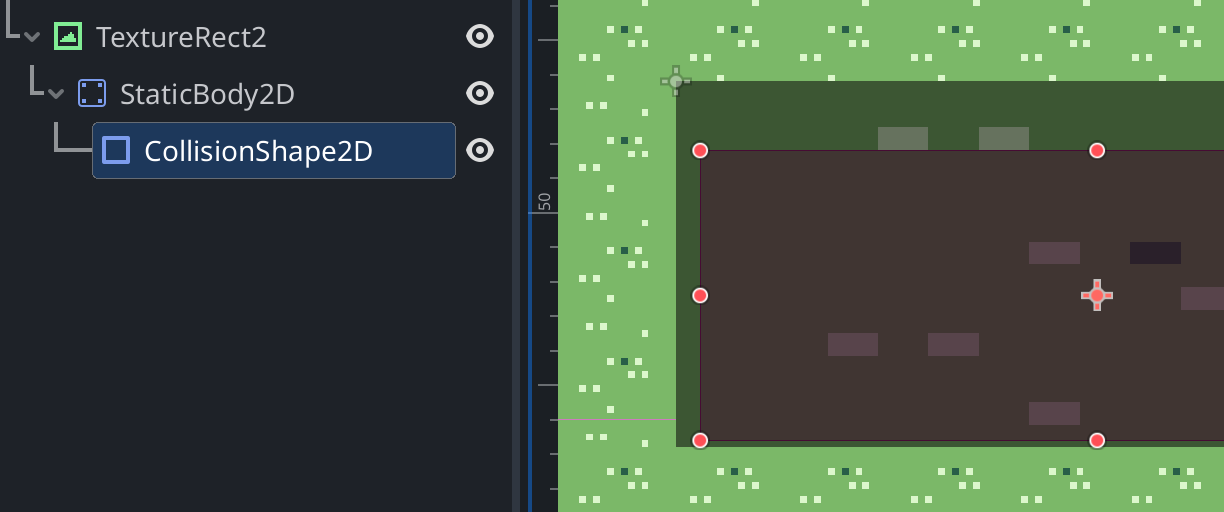
Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Collision areas (Hindernisse)

* Benötigt drei Nodes (siehe screenshot)
* TecxtureRect wird wieder ein PNG File hinzugefügt ... ‚ z.B. green\_tile.png‘ im Stretch Modus ‚Scale‘

Im Inspector kann die Farbe des Hindernisses dann noch per ‚modulate‘ verdunkelt werden



# Vervielfachen von Playern, Hindernissen, etc.

**Welche Ressourcen werden geteilt?**

var tex = preload("res://icon.png") # tex ist eine Resource (Texture2D)

var img = TextureRect.new() # img ist ein Control (Node)

img.texture = tex # Resource wird in Node verwendet

add\_child(img)

* Ressourcen wie Texture2D oder Shapes werden geteilt per Referenz auf das selbe Objekt
* Verhaltens oder Control Nodes werden instanziiert

## Node- oder Szeneneigenschaften per @export verändern

z.B. Größe ändern (entweder statisch im Editor oder später als ‚game mechanic‘

@onready var sprite\_2d: Sprite2D = $Sprite2D

@export var sprite\_2d\_scale: Vector2 = Vector2(1, 1)

func \_ready():

sprite\_2d.scale = sprite\_2d\_scale

**Warum ist das notwendig?**

@export darf nur **feste Initialwerte** enthalten, keine Funktionsaufrufe oder Node-Zugriffe.

$Sprite2D.scale wird zur Laufzeit evaluiert – @export wirkt aber nur zur Kompilierzeit.

## Szenen per Code vervielfachen

var player\_scene = preload("res://Player.tscn")

var player\_a = player\_scene.instantiate()

player\_a.speed = 600.0

var player\_b = player\_scene.instantiate()

player\_b.speed = 300.0

add\_child(player\_a)

add\_child(player\_b)

# Annex

**Hintergrundwissen:** Zwei Wege um ‚Gravity‘ zu ändern ...

Quelle <https://www.youtube.com/watch?v=trImNx6vWwY&ab_channel=NADLABS>

* global gravity change under Projekteinstellungen / Physik / 2D (für alle Objekte)
* avoid gravity in func \_process() with
  + velocity.y = 0
  + move\_and\_slide()

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

Automatisch generierte Beschreibung