Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Ziel der Übung

Es geht um die Simulation von Dialogen und den sich daraus ergebenden Konsequenzen im Spielgeschehen. Konsequenzen können Dialogverläufe oder aber auch Punkte für ‚social skills‘ sein. Punkte können sich dann wieder auf konkretes Verhalten bzw. Darstellungen der Map etc. auswirken.

Im Vorliegenden Beispiel kann der Dialog per ‚Enter Key‘ gestartet werden, vorausgesetzt der Charakter befindet sich in der Nähe eines dialogfähigen Objektes / NPCs.

**Start und Endzustand des Projektes**

Ein Bild, das Text, Screenshot, Schrift, Electric Blue (Farbe) enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Verwendete Konzepte

* **Add-Ons:** Nutzung eines Add-Ons aus dem Godot Ökosystem ‚Dialogue Manager 3‘
* **Actionable** Objekte, stellt Funktionalität dar, wie z.B. die Möglichkeit einen Dialog zu starten. Das Actionable Objekt kann dann jedem beliebigen Element im Spiel (Charakter, Baum, NPC etc.) zugeordnet werden.
* **Layer-Mask-Ebene:** Äquivalent zum Actionable gibt es dann auch ein ActionableFinder Objekt. Beide Objekte können per Layer und Mask eindeutig zugeordnet werden (z.B. im Beispiel Layer 5). Dadurch können auch komplexere Welten geschaffen werden, in denen ein NPC unterschiedliche Dialoge führen kann, je nach dem auf welcher ‚Layer-Mask-Ebene‘ er/sie angesprochen wird.
* **Finetuning**: Detektion eines Mini-Bugs, wie kann ich die Bewegung des Charakters von der Bewegung zwischen den Antwortoptionen im Menü trennen? Beides wertet die selben Tastatureingaben aus. (Spoiler: Entdecken eines Signals im Add-On)

**Vorschau der Projektstruktur:**

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

KI-generierte Inhalte können fehlerhaft sein.

# Vorgehen

## Dialog Add-on installieren

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

KI-generierte Inhalte können fehlerhaft sein.

**Benötigtes Add On:** Für die Implementiernung der Dialoge wird ein ‚Add on‘ von Godot verwendet ...

1. im ‚AssetLib‘ nach ‚Dialogue Manager 3‘ suchen und installieren
2. unter *Projekteinstellungen / Plugins* das Add On aktivieren; unter *Projekteinstellungen / Autoload* sollte es automatisch erscheinen
3. das Plugin ist aktiv in der Weiterentwicklung (rechts oben gibt es dann einen Verweis auf mögliche Updates)
4. Nach erfolgreicher Installation gibt es im Godot-Dateisystem einen neuen Ordner ‚addons‘

🡺 Siehe Dokumentation für mögliche Dialog-Interaktionen und Syntax ... <https://github.com/nathanhoad/godot_dialogue_manager/tree/main?tab=readme-ov-file>

* Hier können auch Verzögerungen in der Textausgabe eingebaut werden oder Textanimationen
* Oder Antworten werden per Zufallswahrscheinlichkeiten entschieden, z.B. ein Hilfsangebot kommt nur in 20% der Fälle, in denen ein NPC angesprochen wird
* Oder ein Dialog wir an Bedingungen geknüpft <https://github.com/nathanhoad/godot_dialogue_manager/blob/main/docs/Conditions_Mutations.md>

🡺 Ein **Beispiel Dialog** findet ihr im Ordner ‚dialogue‘ des finalen Projektes. Die pinken Überschriften sind ‚chapter title‘ und dienen als Sprungmarken; d.h. es könnte auch Sprungmarken geben wie ‚Szene\_01‘ / ‚Szene\_02‘/ etc. Beim Aufruf eines Dialoges kann ich somit den konkreten Startpunkt eines Dialoges definieren.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

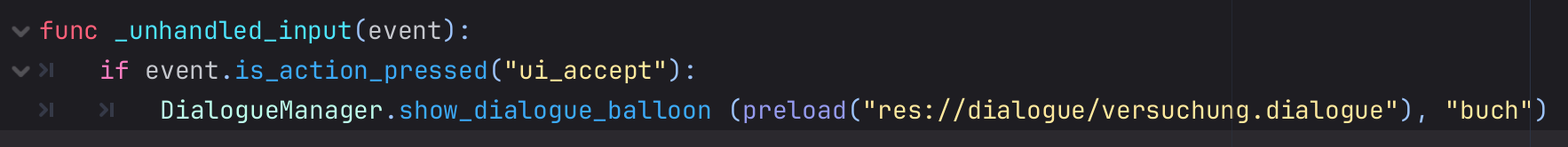
## Einen ersten Dialog erstellen und testen

Im Add-on können Dialoge erstellt und getestet werden ..

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

KI-generierte Inhalte können fehlerhaft sein.

Zum testen

1. Test Szene im Add-on
2. Direkt im Projekt / Charakter Script testen (das meiste sollte sich per ‚Auto-Complete‘ anbieten)   
   
3. Unter Projekt / Tools / Create Copy ... kann ich die Dialogausgabe vom AddOn in mein Projekt kopieren und somit verändern. Den entsprechenden Ordner habe ich ‚dialogue\_style‘ genannt. Optionaler Vorteil: Dialog Font Size ändern, z.B. 40 pixel

* An dieser Stelle wird schon das Problem sichtbar, dass Charakterbewegung und Bewegung im Menu getrennt werden müssen

## Actionable: NPCs oder Objekte werden dialogbefähigt

* Hierfür wird ein ‚Actionable‘ (siehe Abschnitt ‚verwendete Konzepte‘) auf Basis eines Area2D Nodes erstellt
* Das Actionable wird erst später mit entsprechenden Szenen verlinked, daher wandert ‚actionable.tscn‘ in einen ‚helpers‘ Ordner
* Wichtig ist hier schon, ein eigenes ‚Collision Layer‘ (z.B. 5) anzulegen und entsprechend zu benennen (Actionables)

Damit das Actionable universell einsetzbar wird, braucht es eine Skripterweiterung (actionable.gd) :

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Jetzt kann das Actionable mit beliebigen Nodes verlinked werden, um sie dialogfähig zu machen ... siehe im Beispiel – das Buch und Bier Sprite2D

* Hier noch die CollisionShape2D hinzufügen!

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

KI-generierte Inhalte können fehlerhaft sein.

Als letztes muss noch die Dialog Ressource und der Dialog Start spezifiziert werden (jeweils in Abhängigkeit des dialogisierenden Objektes:

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

KI-generierte Inhalte können fehlerhaft sein.

## ActionableFinder: Character kann

Zusätzliche Funktionen im Sprouty Script:

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

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Angepasste Speech Bubble (z.B. Character Image)

* Kopiere die Dialog-Szene aus dem Add-On (Project / Tools / Create Copy of Dialogue Example Baloon) in den Asset Ordner, z.B. unter ‚dialogue\_bubble‘
* Unten zeigt sich die kopierte Struktur von Sprechblase (Balloon) und Dialog Optionen (Responses)

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

KI-generierte Inhalte können fehlerhaft sein.

1. im ‚Dialogue‘ Node wird ein HBoxContainer eingefügt und im HBoxContainer noch ein TextureRect für das Image, wichtig für den VBoxContainer:

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

KI-generierte Inhalte können fehlerhaft sein.

* TextureRect, das TextureRect sollte ein Minimum Size von 150px haben
* HBoxContainer sollte einen ‚separation value‘ (Abstände) von 30 haben

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

KI-generierte Inhalte können fehlerhaft sein.

1. Balloon Script bearbeiten (optional)

In balloon.gd



In func apply\_dialogue\_line() -> void:

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

KI-generierte Inhalte können fehlerhaft sein.

In actionable.gd muss dann nicht mehr der ‚example balloon‘ aufgerufen werden, sondern der eigene balloon

Ein Bild, das Text, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Quellen

**Basis:** Beginner dialogue tutorial for Godot 4

<https://www.youtube.com/watch?v=UhPFk8FSbd8&ab_channel=NathanHoad>

Dialogue Manager einbinden

<https://www.youtube.com/watch?v=UhPFk8FSbd8&ab_channel=NathanHoad>

Installieren und kurze Intro zu spezifischen BB Code Features ("Bulletin Board Code")

<https://www.youtube.com/watch?v=Ydzj1bT_pC8&ab_channel=NathanHoad>

Custom speech bubble einbinden ...

<https://www.youtube.com/watch?v=X0e-n7dbff8&ab_channel=NathanHoad>

Angepasste speechbubble

Part I <https://www.youtube.com/watch?v=X0e-n7dbff8&ab_channel=NathanHoad>

Part II <https://www.youtube.com/watch?v=DfdcyHwqXdo&t=0s&ab_channel=NathanHoad>

(Teil II enthält Korrekturen ! )

Developers itch.io page

<https://nathanhoad.itch.io/>