Game Signale über einen Event-Controller auswerten   
(Szenenübergreifend)

Ein Bild, das Screenshot, Diagramm, Plan enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Ziel**: Mit Hilfe eines Event Controllers wirken sich Signale nicht nur auf den aktuellen Zustand der beteiligten Nodes aus (Blumen und Pilze, Character), sondern können (fast) jede andere Darstellung im Spiel beeinflussen. So können Barrieren verschwinden oder entstehen oder User Interface zeigt die aktuell gesammelten Ressourcen an etc.

**Start und Endzustand des Projektes**

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

KI-generierte Inhalte können fehlerhaft sein.

# Signale können abonniert werden (mittels Event Controler)

Aufgabe des Event-Controller ist vor allem das *Verteilen* und *Koordinieren* von Signalen*.* Die Reaktionslogik findet dann in den betroffenen Szenen statt. Durch Signale und Event-Controller erhält das Projekt eine **ereignisgetriebene Architektur.**

Signale ‚verschlanken‘ die Frames (nur tatsächlich ausgelöste Events werden verarbeitet). Somit braucht das Spiel weniger Hardware Ressourcen etc. Ein Frame ist jede Iteration des Game-Loops (→ Update → Physik → Render).

Auch entscheidend für *inklusives Design:* zusätzliche Listener (z. B. für Screenreader-Ausgaben, haptisches Feedback oder alternative Eingabegeräte) können ohne Eingriffe in Kernsysteme angebunden werden – ein essenzieller Faktor für eine bessere Zugänglichkeit.

## Wie implementiere ich Abonnement und Event Controller?

5 Schritte:

1. **Trigger programmieren (z.B. Auge triggert Signal)**

* Objekt ‚EventController‘ wird erst in den nächsten zwei Schritten definiert

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

KI-generierte Inhalte können fehlerhaft sein.**

1. **EventController für Signale definieren**
2. Signale werden im **Singleton 'event\_controller.gd'** definiert ...

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

KI-generierte Inhalte können fehlerhaft sein.

1. Singleton wird als **global verfügbare Autoload-Ressource** eingerichtet

Singleton in Projekteinstellungen / Globals / Autoload eintragen und aktivieren.

Hintergrundwissen:

* *Script‐Datei* (**event\_controller.gd**) = Quelltext, der eine Klasse definiert.
* *Node‐Instanz* (**EventController**) = zur Laufzeit erzeugtes Objekt dieser Klasse, das Teil des Szenenbaums oder – bei Autoload – Teil der Root‐Ebene ist.

1. **Signal senden**

* als Autoload kann das neue Script / der neue Node 'event\_controller.gd' in jeder Szene verwendet werden, z.B. in **flower.gd**

*EventController.emit\_signal('met\_flower').*

1. **Signal abonnieren**
2. Jede Szene, die das Signal auswerten möchte, muss sich zuerst mit dem entsprechenden Singleton, in ihrer \_ready() Funktion, verknüpfen und braucht dafür zwei Parameter, Signalname und auszuführende Funktion. *EventController.connect("met\_flower", on\_event\_met\_flower)*
3. Signal auswerten: die gerade angeführte Funktion beinhaltet dann die gewünschte Reaktion, z.B.

In folgendem Beispiel, abonniert die ‚World Scene‘ das Signale ‚met\_eye‘, d.h. eine Spielerin ist auf das Auge gestoßen und jetzt kann in der Weltszene ein TeilMapLayer Bereich unsichtbar gemacht werden und die Colllision wird ausgeschaltet (im Spieldesign könnte das also die Öffnung eines Zuganges bedeuten).

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

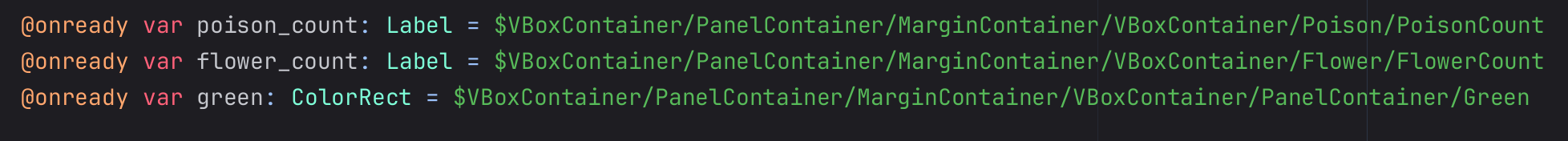
KI-generierte Inhalte können fehlerhaft sein.

**5) @onready verwenden**

Insbesondere wenn das Signal eine Anzeige in der UI ändern soll. kann der Pfad zum zu verändernden Label recht lang sein. UIs sind meist geschachtelte Control Nodes (Kontrolle, i.S. einer Nutzer:innensteuerung).

Wenn ich den gewünschten Node (z.B. ein ‚Label‘ Node für eine gewünschte Anzeige) in eine freie Zeile noch vor die Funktion \_ready(): ziehe und dabei die STRG Taste gedrückt halte, dann sollte eine ähnlich strukturierte Zeile automatisch erscheinen...

Im Folgenden kann ich Gift- und Blumenkontakte hochzählen und mit ‚green‘ kann ich die Länge meines Gesundheitsbalken kontrollieren.



Im konkreten Beispiel muss für die Veränderung des Labels (ein Textfeld) noch das Format angepasst werden, d.h. mein Counter ist eine Zahl und mein flower\_count.text ist eine String / Text. Daher brauche ich die Funktion str() um eine Zahl in einen Text umzuwandeln, bevor ich dann die Zahl im Textformat auch einem Textlabel zuweisen kann.

*func on\_event\_met\_flower():*

*print('UI triggered')*

*counter += 1*

*flower\_count.text = str(counter)*

# UI Szene

Godot unterscheidet zwischen Layers und Szenenbäumen. Innerhalb einer Canvas entscheidet die Baum­hierarchie; zwischen verschiedenen Canvas-IDs entscheidet allein das layer-Feld. Renderer sortiert strictly nach dieser Zahl → Layer 0 zuerst, dann 1, 2 … – unabhängig von Scene-Baum-Position oder Ladezeitpunkt.

🡺 siehe auch HUD Beispiel in ‚99\_Gemüselauf\_Game‘

**Vorteile des CanvasLayer-Ansatzes**

* + **Unabhängig von Kameratransform**: HUD bleibt fixiert, selbst wenn die Kamera zoomt oder rotiert.
  + Jeder Layer bekommt explizite Position zugeordnet (0 = Welt, 1 = HUD, 2 = Accessibility, 3 = Debug).
  + **Barrierefreiheit**: Screen-Reader- oder High-Contrast-Layer ScreenReader\_layer ≥ HUD\_layer+1).

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

KI-generierte Inhalte können fehlerhaft sein. Ein Bild, das Text, Whiteboard, Handschrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

* **Healthbar** verwendet 2 x ColorRect
* Minimum Size im Inspector entspricht der Länge des Lebensbalken (z.B. 200)
* Container Sizing des grünen Balkens wird auf ‘Zum Anfang verkleinern’ gesetzt

# Übung

1. Zur Zeit werden nur die Blumen aber nicht die giftigen Pilze getrackt. Zähle die Pilzkontakte und reduziere den Lebensbalken um jeweils 10%.
2. Erstelle Dein eigenes HUD / UI Layout.

## Weitere Nutzung von Signale

Beispiel

* Spieldauer wird angezeigt: Timer Node sendet ein Signal im Sekundentakt um ein Update der Zeitanzeige zu triggern

🡺 siehe Beispiel ‚99\_Collectibles\_mit\_Timer‘

* Per Escape Taste erscheint dann ‚Pause‘ und die Zeitanzeige bleibt stehen:  
  InputController (ein Autoload Singleton) beobachtet Tastatureingaben und triggert ein Signal sobald die Escape Taste betätigt wird

# Ressourcen

How to Create Animated TileSets in Godot 4

<https://www.youtube.com/watch?v=AO-pqAvzowk&ab_channel=DevDuck>

How To Create Collectible Objects In Godot (Tutorial)

Quelle: <https://www.youtube.com/watch?v=Rh_8UXjYTn4>

# Annex: Background-Wissen / ältere Inhalte

## Wie setzt sich der Kopf einer Signalfunktion zusammen? Beispiel \_on\_area\_2d\_body\_entered(body)

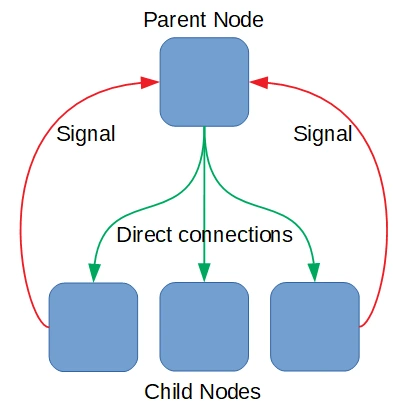
🡺 **Präfix, emittierender Node-Name** und **Signal-Name**

* Das Präfix \_on\_ signalisiert, dass diese Funktion ein **Signal-Callback** ist, d. h., sie wird **automatisch aufgerufen**, wenn ein spezifisches Signal von einem Node ausgelöst wird
* Der Name area\_2d bezieht sich auf Signalsender -Node (könnte auch flame\_area\_2d heißen)
* body\_entered ist der Name des Signals, das vom Area2D-Node ausgesendet wird
* Das Argument body ist eine Referenz auf den Körper (Node), der das Signal ausgelöst hat (typischerweise ein CharacterBody2D, RigidBody2D, oder ein anderer PhysicsBody2D). Der Node-Name kann dann auch per print(body.name) abgefragt werden.

## Lose und enge Verknüpfung

Unterschied zwischen loser und enger Verknüpfung zwischen Signalen und Knoten.

... siehe auch <https://gdscript.com/solutions/signals-godot/>:



## Typische Singletons

* *Game Controller* (z.B. Spielerstatus wie Gesundheit, Inventar abspeichern
* *Audio Controller* (z.B. Wechsel der Hintergrundmusik, Lautstärkeregelung, Soundeffekte wie Schritte, Einstellungen im Audio Controller bleiben über Szenen hinweg erhalten)
* *Input Controller* (z.B. Pause, Mini-Map, Erklärungen, Inventarübersicht anzeigen)

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

Automatisch generierte Beschreibung