

# Workshopopdracht: Health Potion maken in Godot

## Doel van de opdracht

In deze korte workshop maak je een health potion die de speler kan oppakken. Je leert:

- Wat nodes en scenes zijn
- Hoe je een sprite met animatie toevoegt
- Wat variabelen en functies zijn
- Hoe je een simpel script schrijft
- Hoe je collision detectie gebruikt (Area2D)
- Hoe je geluid afspeelt

**Belangrijk: Als je vast zit, vraag om hulp!**

## Het project verkennen

1. Open het project in Godot
2. Start de game door op de playknop te drukken rechtsboven of F5
3. Beweeg met WASD val aan met linker muisknop
4. Stop de game door op het kruisje van de opgekomen window te klikken of F8

Je ziet dat er al een werkende game is met een speler en platforms. Jij gaat nu een health potion toevoegen!

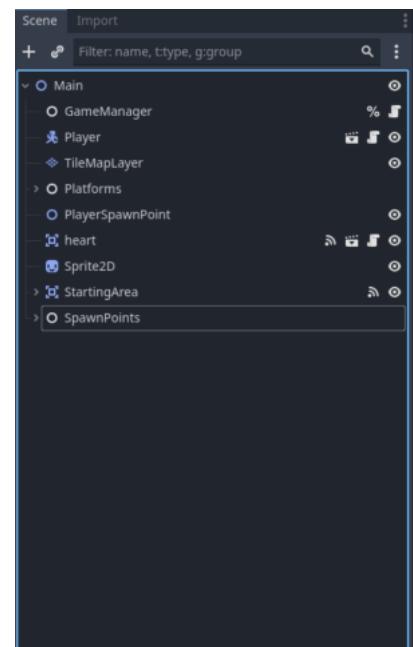
## Wat zijn nodes en scenes? (Theorie)

Bekijk aan de linkerkant van je scherm de **hiërarchie** (Scene Tree). Hier zie je de onderdelen die ik heb toegevoegd voor de aanwezige functionaliteit.

In Godot bestaat een game uit **nodes**. Een node is een bouwblok met een specifieke functie.

Voorbeelden van nodes:

- **Node2D**: Heeft een positie op het scherm (x, y coördinaten)
- **Sprite2D**: Toont een afbeelding
- **Area2D**: Detecteert wanneer iets in een bepaald gebied komt
- **AudioStreamPlayer2D**: Speelt geluid af



## Waarom is dit handig?

Godot heeft deze nodes al voor je gemaakt, zodat je niet alles zelf hoeft te programmeren. Je combineert verschillende nodes om complexere dingen te maken.

## Nodes kunnen andere nodes bevatten

Bijvoorbeeld, een HealthPotion kan bevatten:

- Een AnimatedSprite2D (voor de visuals/animatie)
- Een Area2D (voor detectie wanneer speler erin loopt)
- Een CollisionShape2D (om de vorm van de detectie te bepalen)
- Een AudioStreamPlayer2D (voor een geluidje)

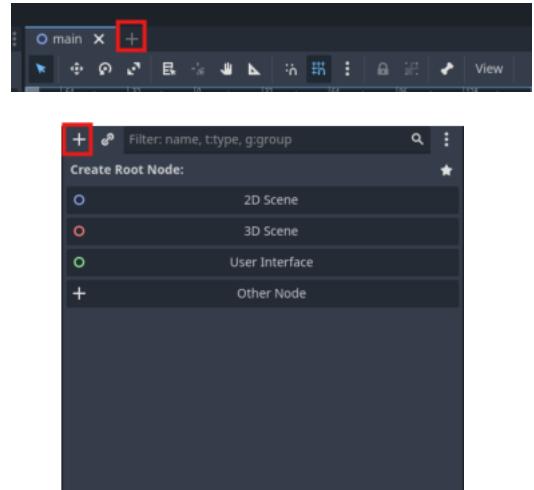
## Wat is een scene?

Een collectie van nodes kun je opslaan als een **scene**. Dit is een herbruikbaar onderdeel. Als je 10 potions wilt, maak je 1 potion scene en hergebruik je die 10 keer!

## Stap 1: Een nieuwe scene maken

Nu gaan we onze eigen health potion scene maken!

1. Klik linksboven op het **plusje** naast "Main" (New Scene)
2. Klik op het **plusje** in de Scene Tree (of druk Ctrl + A)
3. Zoek naar **Node2D** en klik **Create**
4. Hernoem de node naar **HealthPotion** (dubbelklik erop)
5. Sla de scene op (Ctrl + S):
  - o Map: scenes/
  - o Naam: health\_potion.tscn



## Waarom Node2D?

Dit is de simpelste node met een positie op het scherm. Perfect voor iets dat stilstaat zoals een potion!

## Stap 2: Een sprite met animatie toevoegen

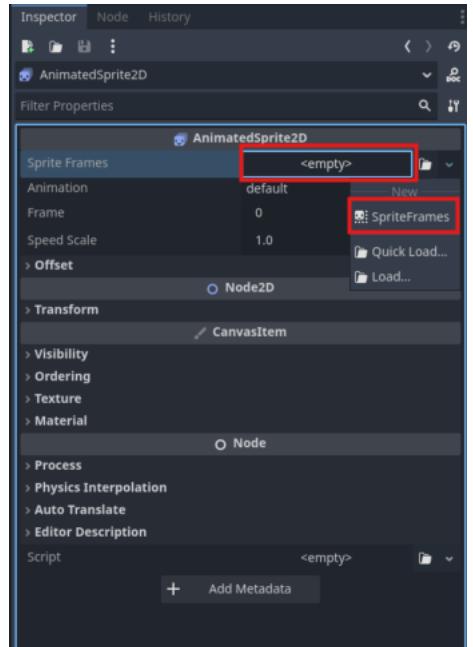
Nu gaan we de potion zichtbaar maken met een animatie!

### De AnimatedSprite2D toevoegen

1. Selecteer de **HealthPotion** node
2. Voeg een **AnimatedSprite2D** toe (Ctrl + A)
3. Selecteer de **AnimatedSprite2D** node

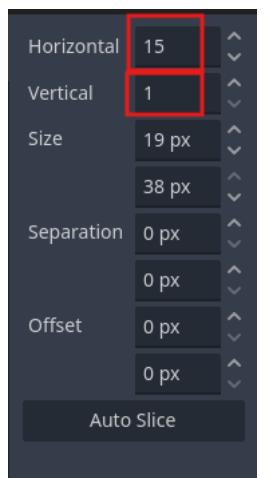
4. In het **Inspector** (rechts):

- Bij **Sprite Frames** → klik <empty> → **New SpriteFrames**
- Klik nogmaals op **Sprite Frames** (onderkant scherm opent)



### Animatie instellen

5. Hernoem "default" naar **idle**
6. Klik op het **grid icoontje** ("Add frames from sprite sheet")
7. Kies één van deze opties uit "assets/sprites/potions" folder
8. Zet vertical op **1**
9. Zet horizontal op hoeveel frames (plaatjes) er zijn
10. Selecteer alle frames (sleep van links naar rechts)
8. Klik "**Add ... frame(s)**"
9. Zet **Loop** aan, als het goed is staat deze al aan (blauw icoontje - de animatie blijft herhalen)
10. Zet **Autoplay** aan (links van Loop - start automatisch)
11. Zet **FPS** op **10.0** (of test wat je mooi vindt!)



### Wat is FPS?

FPS staat voor Frames Per Second - hoeveel plaatjes per seconde. Hoe hoger het getal, hoe sneller de animatie. Zo maak je dus van een aantal plaatjes een animatie!

**Test!** Klik op de playknop in je animatievenster - je zou de animatie moeten zien draaien! 🎉

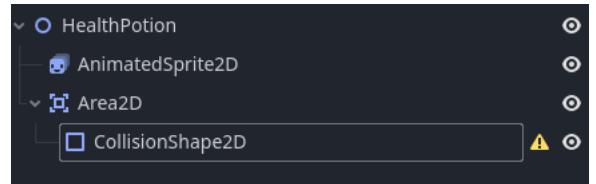
## Stap 3: Collision detectie toevoegen

Nu moet de potion weten wanneer de speler hem aanraakt.

Collision detectie is hoe de game weet wanneer twee dingen elkaar raken. Bijvoorbeeld: speler raakt potion, vijand raakt speler, etc. Wij gebruiken de Area2D hiervoor omdat deze een signaal kan geven wanneer er iets in komt (body entered) zoals de speler

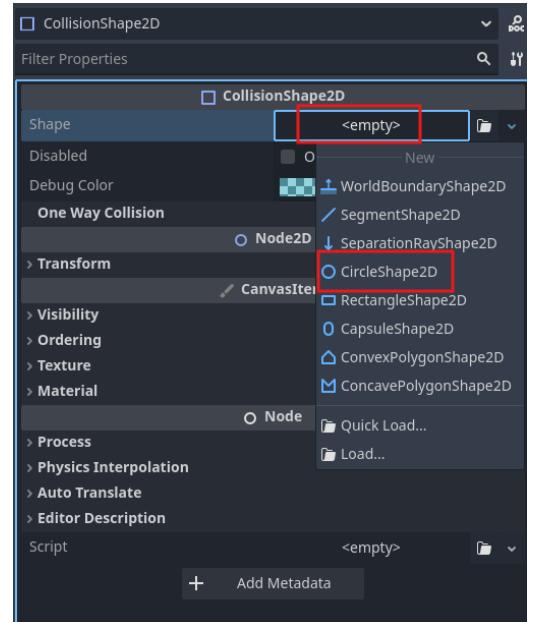
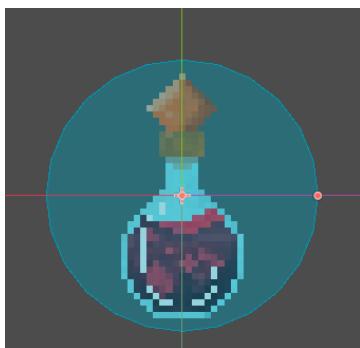
## Area2D toevoegen

1. Selecteer de **HealthPotion** node
2. Voeg een **Area2D** toe (Ctrl + A)
3. Selecteer de **Area2D** node
4. Voeg een **CollisionShape2D** toe als kind van Area2D (Ctrl + A)



## De vorm instellen

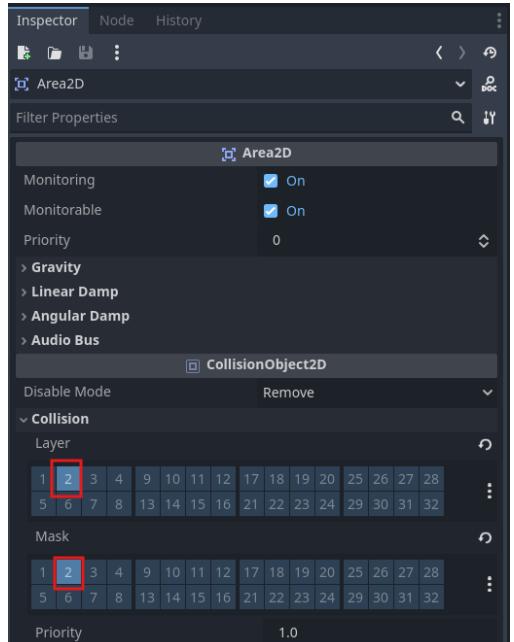
5. Selecteer de **CollisionShape2D**
6. In het **Inspector**:
  - o Bij **Shape** → klik <empty> → New **CircleShape2D**
7. Pas de grootte aan:
  - o Sleep het **oranje bolletje** tot de cirkel om de potion heen past



## Layers instellen (belangrijk!)

De layers bepalen WAT de potion kan detecteren.

8. Selecteer de **Area2D** node
9. In het **Inspector** → **Collision**:
  - o Bij **Layer**: alleen **layer 2** aanvinken, dus 1 uitzetten
  - o Bij **Mask**: alleen **layer 2** aanvinken, dus 1 uitzetten



## Waarom layer 2?

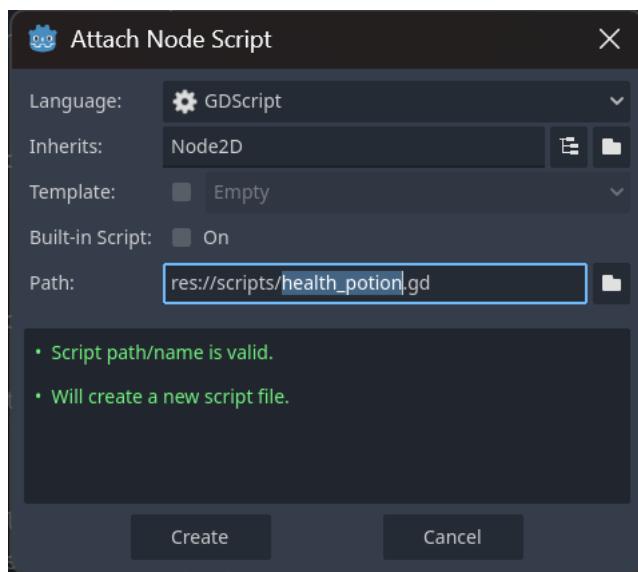
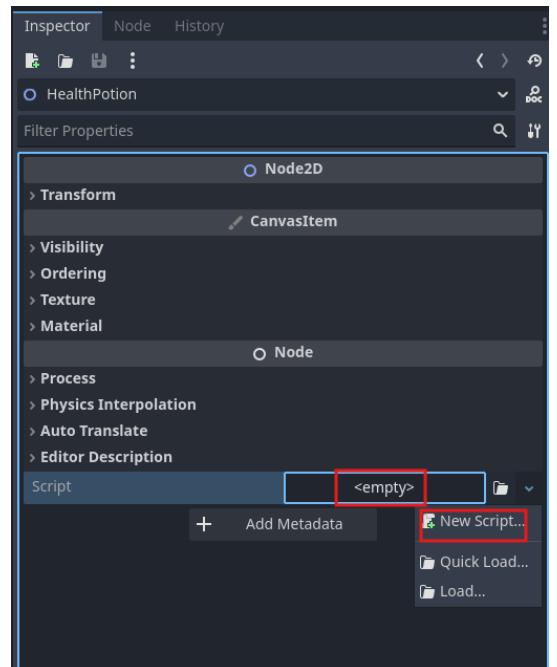
De speler zit op layer 2. Door dit in te stellen, detecteert de potion alleen de speler en niet andere dingen.

## Stap 4: Een script toevoegen

Nu gaan we de logica programmeren!

### Script aanmaken

1. Selecteer de **HealthPotion** node (de root)
2. In het **Inspector**:
  - o Bij **Script** → klik <empty> → **New Script**
3. In het venster:
  - o Klik op het **map icoontje** bij Path
  - o Map: scripts/
  - o Naam: health\_potion.gd
4. Check dat er staat: res://scripts/health\_potion.gd
5. Zorg dat template op Empty staat
6. Klik **Create**



Je ziet nu in je scherm:

```
extends Node2D
```

Dit betekent: "Dit script zit op een Node2D en kan al zijn functies gebruiken."

## Stap 5: Variabelen - Data opslaan

### Wat is een variabele?

Een variabele is een **doosje met een label** waarin je data opslaat. Zoals een doos waar "health" op staat met het getal 100 erin.

### Waarom zijn variabelen handig?

Stel je voor: je wilt weten hoeveel health de potion geeft. In plaats van overal het getal 2 te typen, schrijf je het 1 keer op in een variabele. Later kun je makkelijk veranderen naar 5 of 10!

### Je eerste variabele maken

Voeg dit toe onder extends Node2D:

```
# Hoeveel health geeft de potion?  
var heal_amount = 2
```

### Uitleg:

- Alles achter # is uitleg, dus het heeft geen functionaliteit
- Var geeft aan dat we een variabel gaan maken
- heal\_amount is de naam die wij eraan geven (gebruik hier geen spaties voor, spaties vervangen we met \_ )
- = 2 is de waarde
- Deze variables kunnen we uitlezen door de naam te schrijven, dus heal\_amount. Of aanpassen, naar bijvoorbeeld 5, door heal\_amount = 5 te schrijven

## Stap 6: Functies - Acties uitvoeren

### Wat is een functie?

Een functie is een **stappenplan** dat de computer volgt. Zoals een recept: "eerst dit, dan dat, dan dat."

Godot heeft speciale functies die automatisch worden aangeroepen:

- \_ready(): Wordt 1 keer uitgevoerd wanneer de potion in de game komt

### Je eerste functie schrijven

Voeg deze functie toe onder je variabelen:

```
func _ready():  
    print("Health potion is klaar!")
```

### Uitleg:

- func betekent "hier komt een functie"
- \_ready is de naam
- Alles wat **ingesprongen** staat (met een tab) hoort bij de functie

- `print()` schrijft tekst naar de console (onderaan je scherm)

### **Belangrijk: Indentatie!**

Let op de tabs (inspringen). Godot weet hierdoor wat bij de functie hoort, als je errors (rode gedeeltes) krijgt, dan is dat vaak door indentatie. Dit hoeft je niet te zetten in je code maar is een voorbeeld:

```
func _ready():
    print("Dit hoort bij ready") # ✓ Correct (tab)
print("Dit NIET") # ✗ Fout (geen tab)
```

### **Test het!**

1. Sla op (Ctrl + S)
2. Ga naar de **Main** scene (klik op Main bovenaan)
3. Klik op 2D bovenin
4. Klik op het **kettinkje** (Instantiate Child Scene)
5. Selecteer health\_potion.tscn
6. Plaats de potion ergens in het level
7. Start de game (F5)



Kijk naar de **console** onderaan. Je zou moeten zien:

```
Health potion is klaar!
```

## Stap 7: Signals verbinden - Detecteren wanneer de speler langsloopt

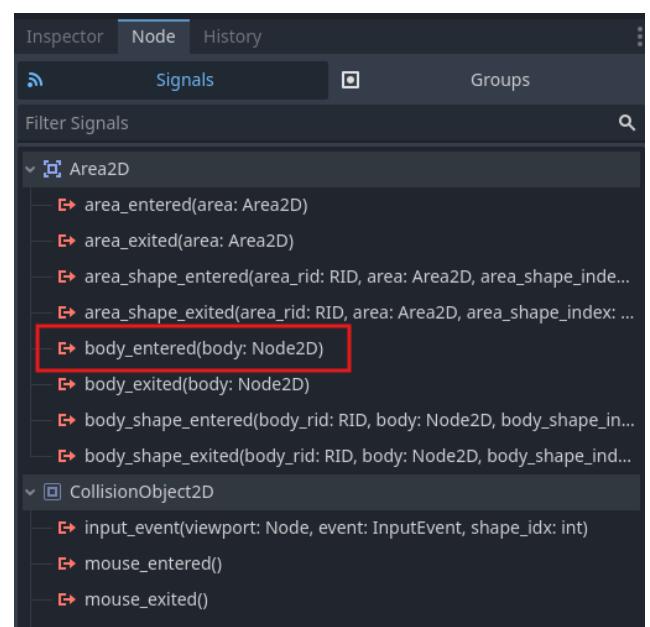
### **Wat is een signal?**

Een signal is een **notificatie** die Godot stuurt wanneer iets gebeurt. Bijvoorbeeld: "Er is iets mijn Area2D binnengekomen!"

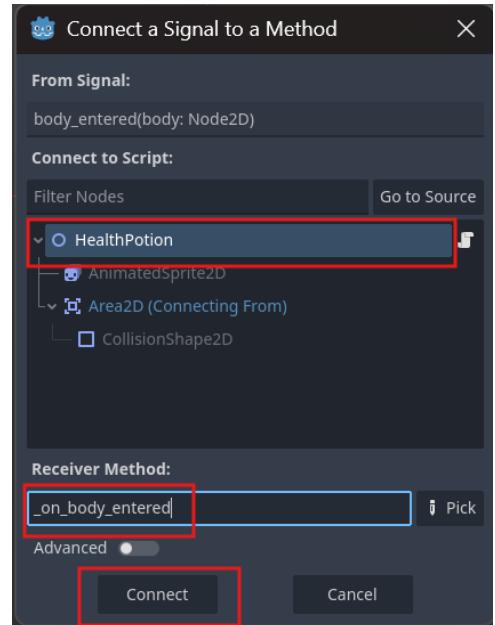
We gaan het `body_entered` signal verbinden. Dit signal wordt verstuurd wanneer iets (zoals de speler) de Area2D binnengaat.

### **Signal verbinden in de editor**

1. Selecteer de **Area2D** node (niet HealthPotion, maar de Area2D eronder!)
2. Klik op het **Node** tabblad rechts (naast Inspector)



3. Je ziet een lijst met signals
4. Dubbelklik op **body\_entered(body: Node2D)**
5. Zorg dat **HealthPotion** geselecteerd is als receiver (gehighlight)
6. Verander de "Receiver Method" naam naar:  
\_on\_body\_entered
7. Klik **Connect**



### **Wat gebeurt er nu?**

Godot maakt automatisch een functie aan in je script! Kijk naar je script, onderaan zie je nu:

```
func _on_body_entered(body: Node2D) -> void:
    pass # Replace with function body.
```

Deze functie wordt aangeroepen **elke keer** dat iets de Area2D binnentreedt!

### **De functie uitbreiden**

Vervang de nieuwe functie met deze code:

```
func _on_body_entered(body: Node2D) -> void:
    # Check of het de speler is
    if body.name == "Player":
        print("Speler raakte de potion!")
```

### **Uitleg:**

- `body: Node2D` is de node die de Area2D binnentreedt (**een parameter**)
- `-> void` betekent "deze functie geeft niks terug"
- `if body.name == "Player":` is een **voorwaarde**: "ALS het de speler is, DAN..."
- Het ingesprongen `print()` gebeurt alleen als de voorwaarde waar is

### **Wat is if?**

`if` betekent "als". Je gebruikt het om keuzes te maken:

- ALS het regent → pak een paraplu
- ALS de speler de potion raakt → geef health

Er zijn veel manieren om `if` te gebruiken, zoals in plaats van `==` (is gelijk), deze te gebruiken: `!=` (is niet), `<` (is kleiner), `<=` (is kleiner of gelijk aan), `>` (is groter), `>=` (is groter of gelijk aan).

Je kan deze condities ook aan elkaar koppelen met de woorden `and` of `or` dus bij `and` moeten allebei zo zijn bij `or` moet eenmaal zo zijn

`else` kan je ook gebruiken na een statement om uit te voeren als de statement niet waar is, je kan deze `else` ook als `elif` schrijven dan kan je nog eens een andere check doen

**Voorbeeld**, hoeft niet in je script:

```
is_alive = true
is_poisoned = false
health = 1

# En (beide moeten waar zijn)
if health > 0 and is_alive:
    print("Leeft nog!")
else:
    is_alive = False
    print("Is dood!")

# Of (een van beide moet waar zijn)
if health == 0 or is_poisoned:
    print("Probleem!")
elif health <= 5:
    print("Niet veel health!")
else:
    print("Helemaal prima!")
```

Dus nu zou hij printen "Leeft nog!" en "Niet veel health!". Je kan zoals je ziet ook true en false als variables waarden hebben, dit noemen we **booleans**.

### Test het!

1. Sla op (Ctrl + S)
2. Start de game (F5)
3. Loop naar de potion met WASD

Kijk naar de **console**. Je zou moeten zien:

Health potion is klaar!

Speler raakte de potion!

Maar de potion verdwijnt nog niet en de speler krijgt nog geen health...

## Stap 6: De speler health geven

Nu gaan we de speler daadwerkelijk health geven!

### De speler heeft al een heal functie

De speler heeft al een heal() functie die ik heb gemaakt. We hoeven alleen maar te checken of die bestaat en hem aan te roepen!

Update je \_on\_body\_entered functie:

```
func _on_body_entered(body: Node2D) -> void:
    # Check of het de speler is
    if body.name == "Player":
        # Check of de speler een heal functie heeft
        if body.has_method("heal"):
```

```

        body.heal(heal_amount)
        print("Speler kreeg ", heal_amount, " health!")

        # Verwijder de potion
        queue_free()
    
```

### Uitleg:

- `body.has_method("heal")`: Check of de speler een functie genaamd "heal" heeft
- `body.heal(heal_amount)`: Roep de heal functie aan met waarde 2
- `queue_free()`: Verwijder deze potion uit de game

### Waarom `queue_free()`?

Dit zegt tegen Godot: "Verwijder deze node veilig aan het einde van het frame." De potion verdwijnt!

### Test het!

1. Sla op (Ctrl + S)
2. Start de game (F5)
3. Val een vijand aan tot je health laag is (zie linksboven)
4. Loop naar de potion
5. Je health gaat omhoog en de potion verdwijnt! 🎉

## Stap 9: Meerdere potions plaatsen

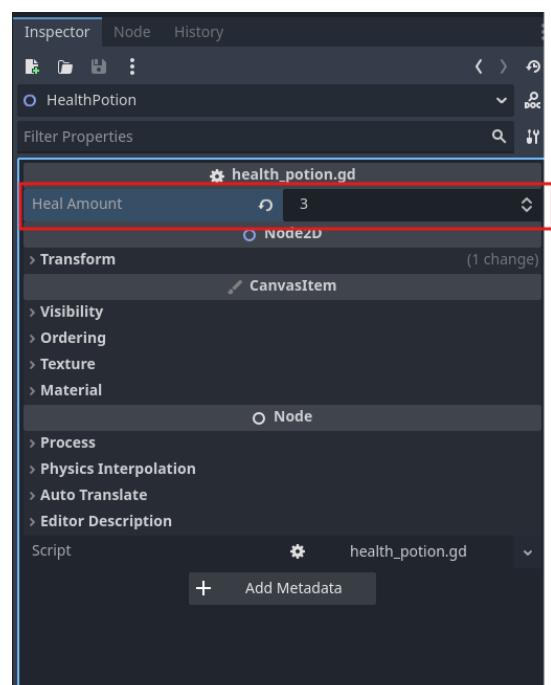
Nu kunnen we gemakkelijk meerdere potions in het level plaatsen!

1. Ga naar de **Main** scene
2. Klik op het **kettinkje** (Instantiate Child Scene)
3. Selecteer `health_potion.tscn`
4. Plaats de potion ergens in het level
5. Herhaal stap 2-4 voor meer potions!

**Dit is de kracht van scenes!** Je maakt 1 potion scene en kunt hem eindeloos hergebruiken. Als je later iets wilt veranderen (bijvoorbeeld meer health geven), hoef je het maar op 1 plek te doen!

Als je wel de `heal_amount` wil veranderen per instance (dus voor elk potion in het level anders), kan je de variable **exporten**. Hierdoor kan je de waarde van deze variable aanpassen in de inspector.

Export de `heal_amount`, vervang dit in je code "var heal\_amount = 2" met dit:



```
@export var heal_amount = 2
```

Als je nu een potion selecteert in je main scene, kan je daar de heal\_amount aanpassen, dit kan dus verschillen per potion in je main scene!

## De complete code

Hier is je volledige health\_potion.gd script:

```
extends Node2D

# Hoeveel health geeft de potion?
var heal_amount = 2

func _ready():
    print("Health potion is klaar!")

func _on_body_entered(body: Node2D) -> void:
    # Check of het de spelers is
    if body.name == "Player":
        # Geef de spelers health
        if body.has_method("heal"):
            body.heal(heal_amount)
            print("Speler kreeg ", heal_amount, " health!")

        # Verwijder de potion
        queue_free()
```

## Wat je hebt geleerd!

- Wat nodes en scenes zijn
- Hoe je een AnimatedSprite2D gebruikt
- Wat een variabele is en hoe je ze gebruikt
- Wat een functie is en hoe je ze schrijft
- Hoe Area2D collision werkt
- Wat signals zijn en hoe je ze verbindt (body\_entered)
- Hoe je een functie aanroeft (heal())
- Hoe je controles doet met if statements
- Hoe je een scene hergebruikt

Klaar! 🎉

Je hebt nu je eerste interactieve object gemaakt! Dit is een basis die je kunt gebruiken voor veel verschillende dingen in je game.

### Volgende stap:

Je kunt nu doorgaan naar de **spike opdracht** (als er minder dan 30 minuten over is) of de **enemy opdracht** (als je meer tijd hebt - deze is uitgebreider).