



Ein Einblick in die GODOT Game Engine





Lesson 02

Some of the most important Nodes and movement



More nodes!!!

Godot Example: Nodes

A few more nodes on the way:

- Timer
- Kinematic Body 2D
- Rigid Body 2D
- Sprite
- Collision shape
- Animation player 2D (will be covered in detail later)

Accessing children of the tree hierarchy in code:

Use \$ to access a child node (f.ex .
\$MyNode, \$Button)

Use . to access specific properties (f.ex.
\$Button.text)

Change visibility of a node:
\$MyNode.show()
\$MyNode.hide()

Export Expert

https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_exports.html

```
export var number = 5
export(int) var number
export(Texture) var character_face
export(PackedScene) var scene_file
export(String, "Thief", "Wizard", "Hexblade") var character_class
Etc...
```

Aufgabe:

- Mach eine 'export variable' checkbox (tip: true/false ist eine boolean)
- Füge zwei Buttons ein, die verschiedenen Text ausgeben (Button1 und Button2)
- Wenn die checkbox an ist, lass zur runtime Button1 erscheinen aber nicht Button zwei, wenn die checkbox aus ist dasselbe umgekehrt



Player movement

"Just make sure that the moment to moment feels good so that when someone's just sitting there with a controller, the room could be empty but they can move their character around - make *that* feel good."

— Noel Berry

Interview with Game Maker's Toolkit, 31 Jul 2019
<https://youtu.be/yorTG9at90g>



First Gamejam:
Kettle game – An great example of bad
platformer controls – why they are bad
and how to make them better

<https://pr0crastigam3s.itch.io/wqj196-strangerecipe-kettle>

How would you improve on them?

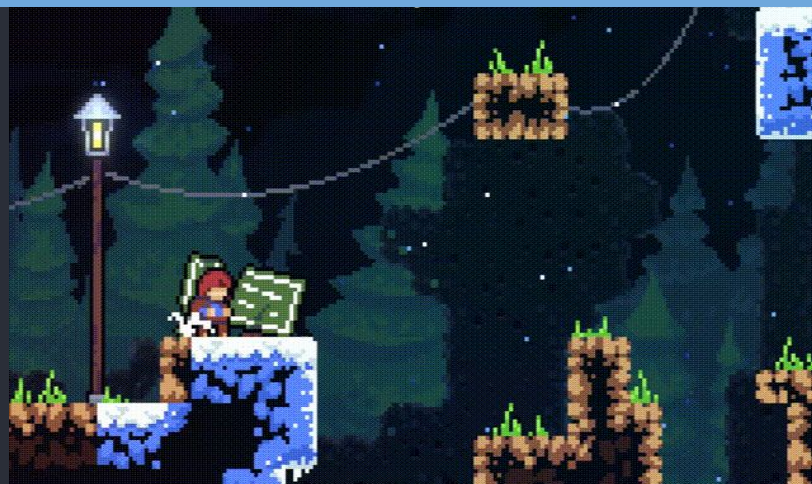


Gut designt ist gut gelungen



Touhou

Die Hitbox des Spielers und der Bullets sind kleiner als der Sprite; Nur der weiße Kreis zählt als getroffen.



Celeste

"Coyote time"
Ein paar Frames nach Verlassen der Plattform kann man noch springen.

Spielespaß im Vordergrund!!!!



Interaktive Essay: Platformer-Toolkit

<https://gmtk.itch.io/platformer-toolkit>



Was könnte man an der Steuerung ändern für:

- Ein Auto mit drift
- Eine Rakete
- Einen Schlittschuhläufer
- Einen Golf Ball
- Einen Schleim
- Einen Vogel
- Einen Würfel
- Eine scheibe Toast
- ???

Ein Problem, viele Lösungen

Nicht "DER" eine richtige Weg
um PlayerControls zu
gestalten

Nicht "DIE" richtige Lösung.
"Für jede Lösung ein Problem"

Let's talk shrimps



Basic Movement (Explained with Shrimps)

This is a northern prawn.

Despite being named "*Pandalus borealis*", it has little to do with pandas.

However, I found it very useful, simply due to the fact you wouldn't expect a shrimp*. It will be your trusty companion sprite for testing movement.

Find it in the Github under

> Assets



*The author of these slides does not take any responsibility if you were expecting a shrimp.

The author just found over the years that a touch of semi-unexpected nonsensicality makes it easier to remember things.

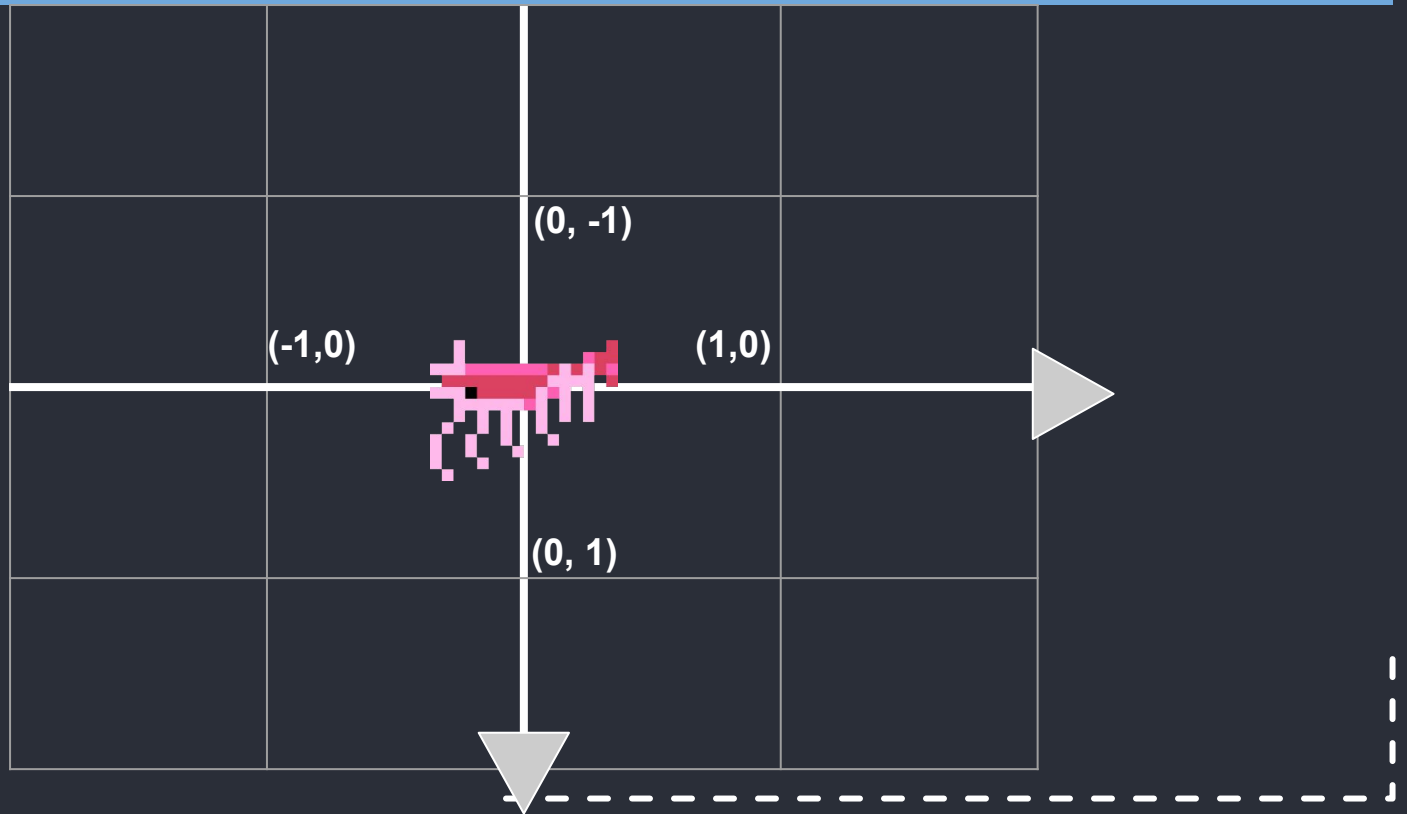
If you indeed DID expect a shrimp: Why???

Ursprung

Koordinaten geben die Position auf dem Bildschirm an. Dein Koordinatensystem fängt oben links an!

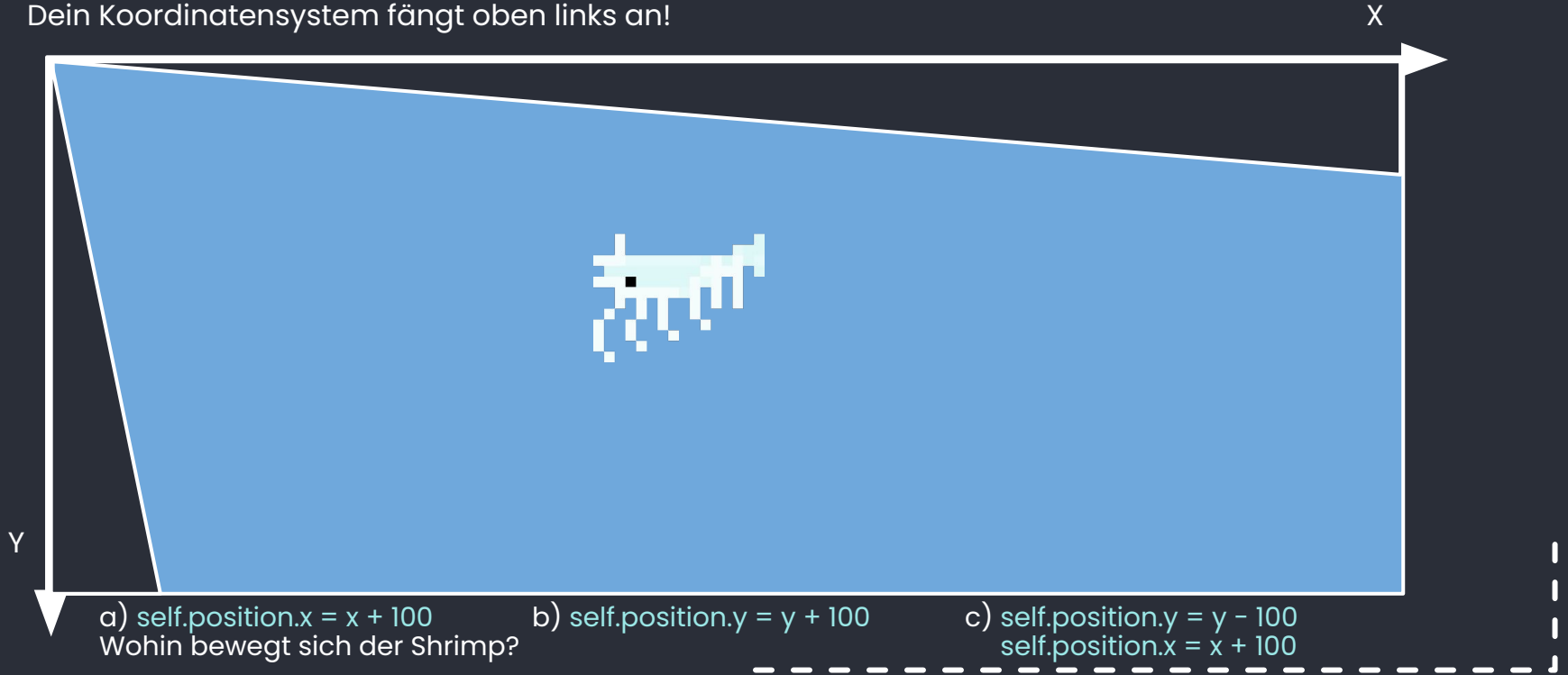


Vektoren



Ursprung

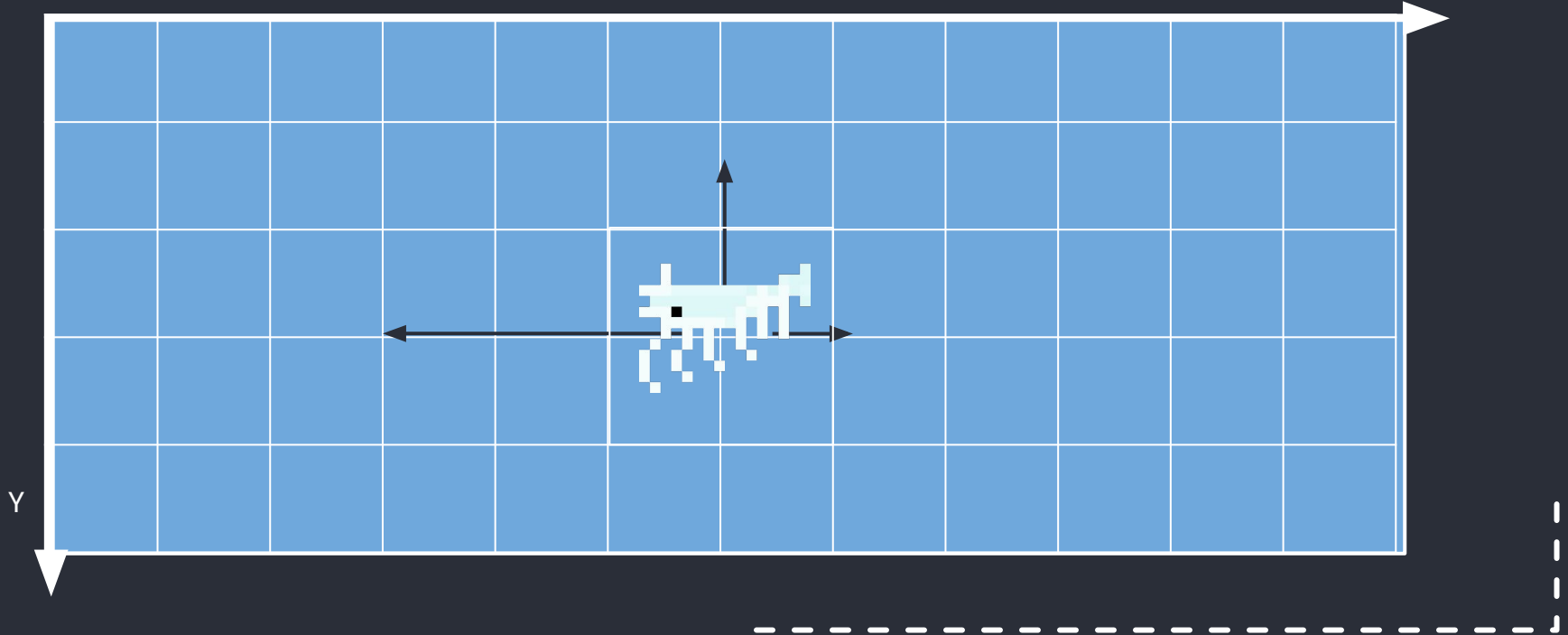
Dein Koordinatensystem fängt oben links an!



Ursprung

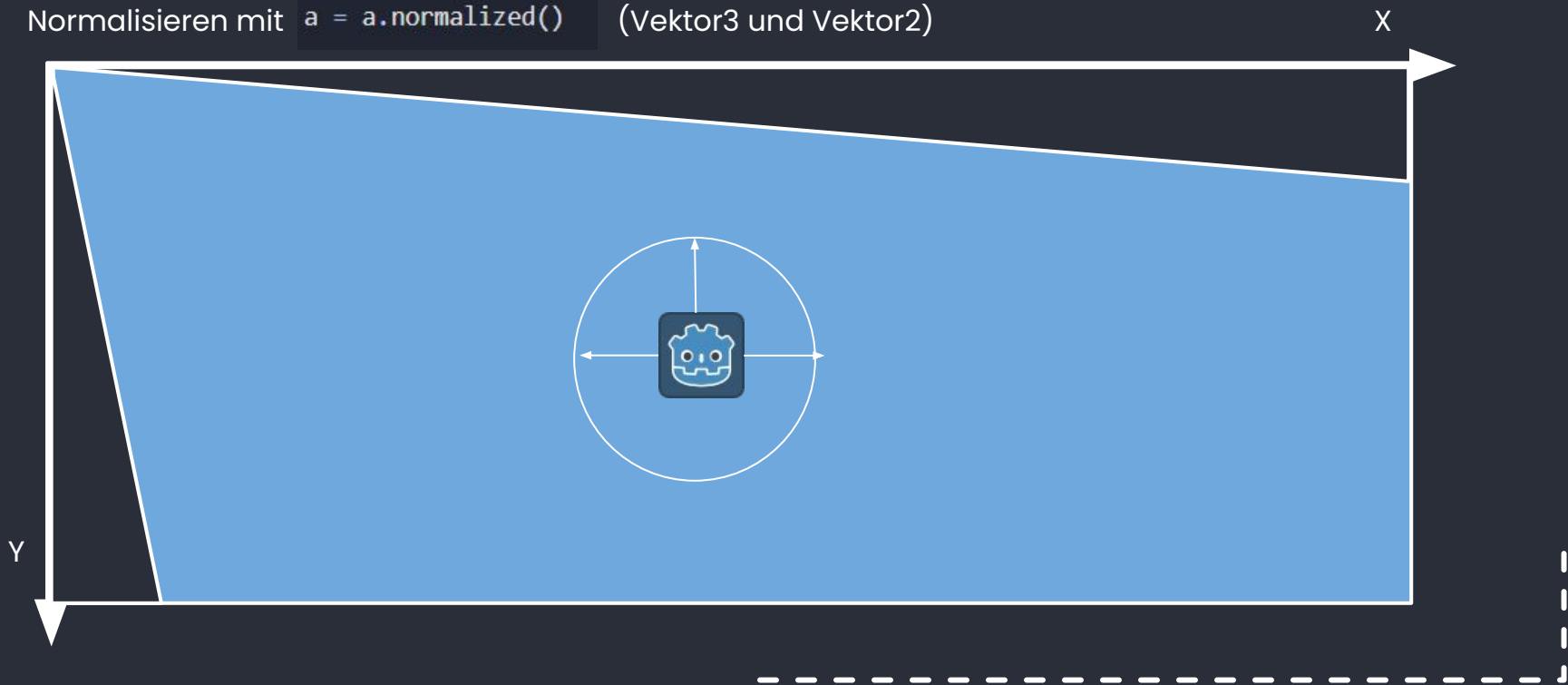
`Vector2()`; Beinhaltet x und y wert. (z.B. `var direction = Vector2(-3, 2)`)

(Documentation: https://docs.godotengine.org/en/stable/tutorials/math/vector_math.html)



Ursprung

Normalisieren mit `a = a.normalized()` (Vektor3 und Vektor2)





Exemplary Shrimp: **Make a shrimp that moves on the screen from left to right!**

Called every frame: `func _process(_delta)`

`Var x = x+1`

Example: Shrimp01



not !

Boolean: Use not or !

By using not (or alternatively !), you can check whether a boolean is not true, or set a boolean to its reverse state. **Booleans can have a state of either true/false or 1/0**

Examples:

if (x == !true): OR if (x == not true):

pass

#asks whether x is false

var b = !true #b is false

var c = !0 #c is true

b = !b

#if b is true, it will be set to false. If b is false, it will be set to true

This can be useful to make f.ex. Hidden objects, that only show to the player on certain conditions. Or change sections of dialogue.

But you will also find this useful for some player controls and toggles.

You will need this in one of the example tasks later, so you'll have a concrete idea how this can be.

What does the following code print?

a)

```
1 extends Node2D
2
3 var x = true
4
5 func _ready():
6     print(not x)
```

b)

```
1 extends Node2D
2
3 var x:bool = 0
4
5 func _ready():
6     print(not x)
```

c)

```
1 extends Node2D
2
3 var x:bool = false
4
5 func _ready():
6     x = 1
7     print(not x)
```

d)

```
1 extends Node2D
2
3 var x:bool = true
4
5 func _ready():
6     x = not x
7     print(x)
8     x = !x
9     print(x)
```

What does the following code print?

e)

```
✓ func _ready():  
  >|   var x:bool = !false  
  >|   var y = false  
  
  >|   print("x is")  
  
  >|   if (not y):  
    >|     _function_1(x)  
  >|   if (y != false):  
    >|     _function_2(x)  
  
✓ func _function_1(a):  
  >|   print(a)  
  
✓ func _function_2(b):  
  >|   print(!b)
```

Use not or !

By using not (or alternatively !), you can check whether a boolean is not true, or set a boolean to its reverse state. **Booleans can have a state of either true/false or 1/0**

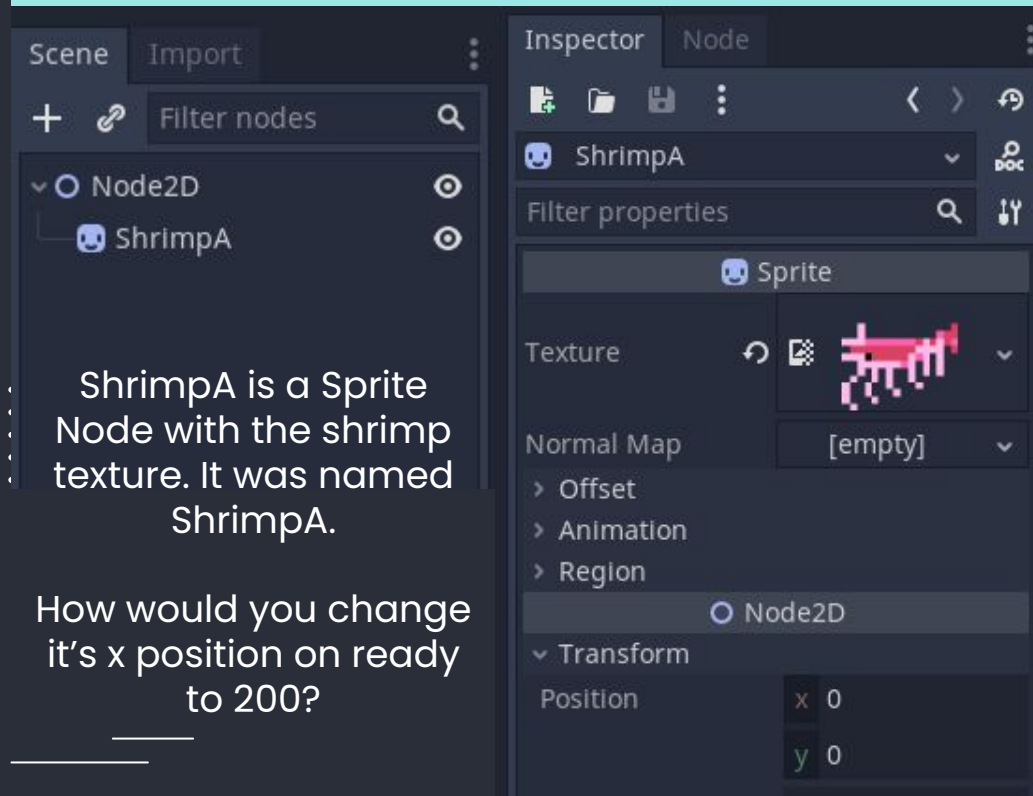
Examples:

if (x == !true): OR if (x == not true):
 pass
 #asks whether x is false

var b = true #b is false

b = !b
#if b is true, it will be set to false. If b is false, it will be set to true

Accessing child nodes/their traits



REMINDER:

Accessing children of the tree hierarchy in code:

Use `$` to access a child node (f.ex. `$MyNode`, `$Button`)

Use `.` to access specific properties (f.ex. `$Button.text`)

Change visibility of a node:

`$MyNode.visible = true`

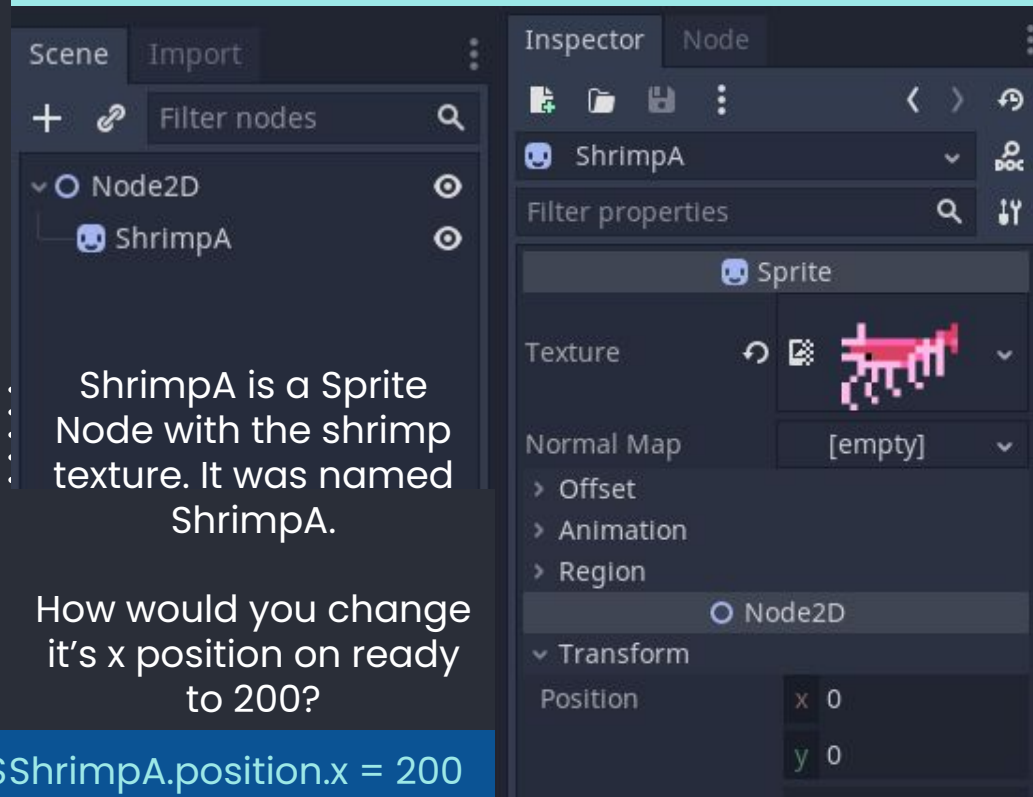
`$MyNode.visible = false`

or:

`$MyNode.show()`

`$MyNode.hide()`

Accessing child nodes/their traits



REMINDER:

Accessing children of the tree hierarchy in code:

Use `$` to access a child node (f.ex. `$MyNode`, `$Button`)

Use `.` to access specific properties (f.ex. `$Button.text`)

Change visibility of a node:

```
$MyNode.visible = true
```

```
$MyNode.visible = false
```

or:

```
$MyNode.show()
```

```
$MyNode.hide()
```

Task 01



- **Make a new 2D node scene, attach a new script to it.**
- **Make a Timer and a Sprite using the Shrimp sprite.**
(The shrimp sprite is available online in the assets folder of the GitHub)
- **Use the Timer signals to make a shrimpA (Sprite) appear after timeout.**
- **(Extra: then change its color after 5 seconds)**

Practice

- **Make a shrimpB (KinematicBody2D) move right across the screen (See exemplary shrimp).**
- **Use two buttons to make shrimpB switch directions on button press, one button left one button right.**
(Extra: Use only one single button)
- **Open the GODOT stable Documentation online. Search for Mouse and input coordinates. Make a shrimp of a different color (using visibility - self modulate) and half size that moves with your mouse cursor.**
- **Make shrimpB face the proper direction it moves in (Looking left moving left, looking right moving right)**
Tip: Access attributes in the inspector by using \$. \$shrimpB.propertyname (f.ex. \$shrimpB.visibility = true).
Play around with the options in the inspector until you find something that would allow you to change its direction. Then try accessing it via code
- **Make another shrimpB (Area2D) in the path of ShrimpA, which starts moving once the previous shrimp collides with it**
Tip: Check out Area2D's built in signals

Physics process engine

○

Rigid Body 2D

Influenced by Physics process engine, can 'fall', gravity and impulses. Don't access it directly

⋮

Static body 2D

Not influenced by physics, not intended to move – but influence Rigid Body 2D

Kinematic body 2D

Not influenced by Physics process engine. All movement is based on your code, but can collide with other bodies – Good for player characters

○

There are specific functions for the Physics process engine (F.ex. `move_and_collide()`)



Exemplary Movement Shrimps

Example: Shrimp02 – 05



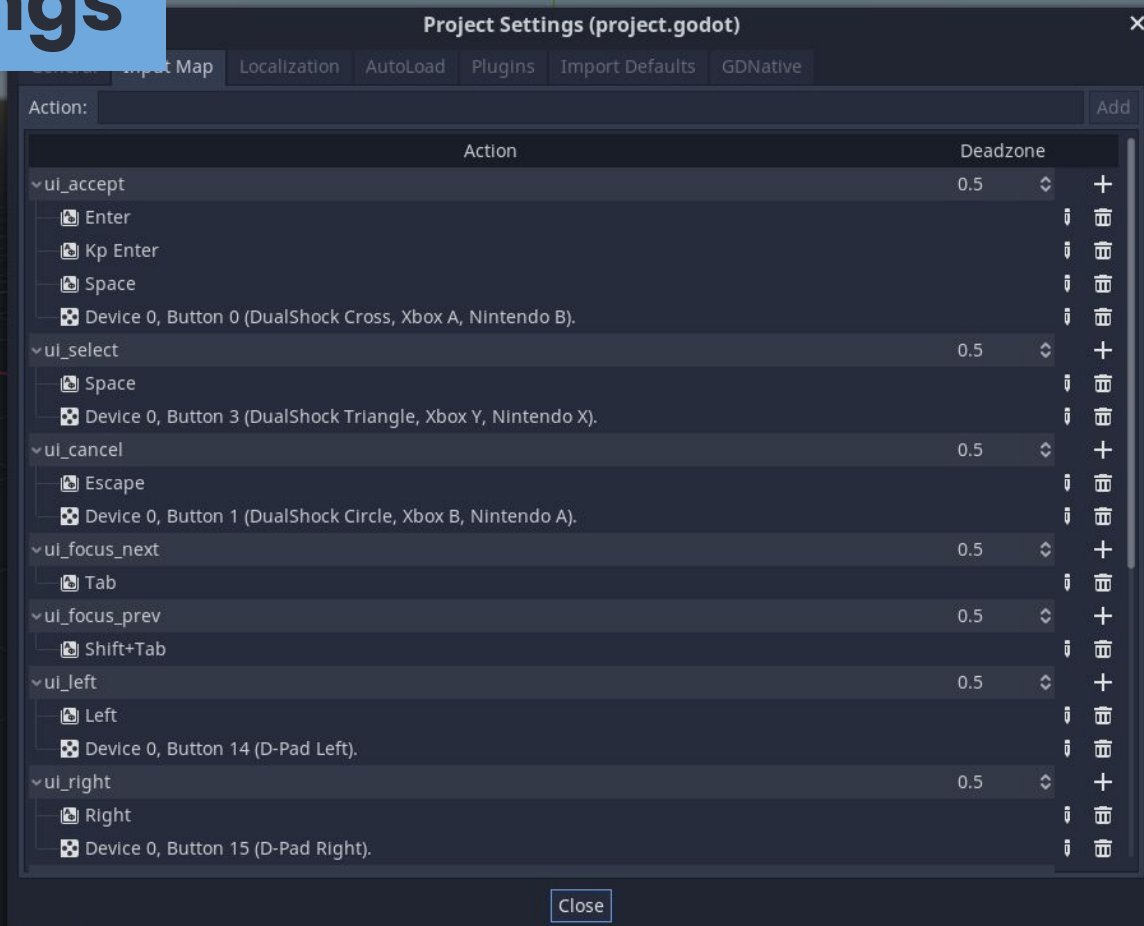
Keyboard settings


Keyboard mapping:

> Project

> Project Settings

> Input Map





**Make a 2D Platformer type jump with your
current knowledge!**

Praxisteil/Selbstaufgabe

