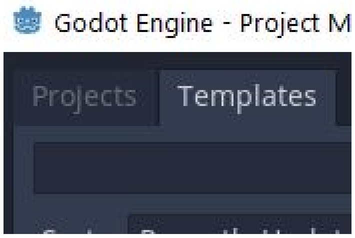
Delicious Spike Engine Help Doc (Written by Sharb aka Renhoex)

I tried to make the engine as user friendly as possible however you may need to delve into the code, as such this guide is meant to be a quick help guide for those new to godot or jumping from engines like game maker.

The most valuable resource you're probably going to need when getting into godot is the official documentation you can find here: https://docs.godotengine.org/en/latest/
The documentation provides just about every definition and feature you'll want to know and provide plenty of tutorials.

If you need examples you can access demos from the launch window under templates



Basics

There is a lot of code functions to cover but I recommend learning these concepts as they have helped me a lot while learning godot

- Delta time (you can lock the game framerate but it's good practice to learn how this works and a lot of the community tutorials will work with delta)
- yield
- node layouts, childs and parents
- Degrees and radians (you'll mostly use deg2rad())
- scenes
- queue_free
- The import menu
- Signals
- Layers and masks
- Animation player (this is going to be your best friend, especially for making complex traps)
- Auto load scripts (These are globals if you're familiar with game maker)
- Tile Maps and Tile sets

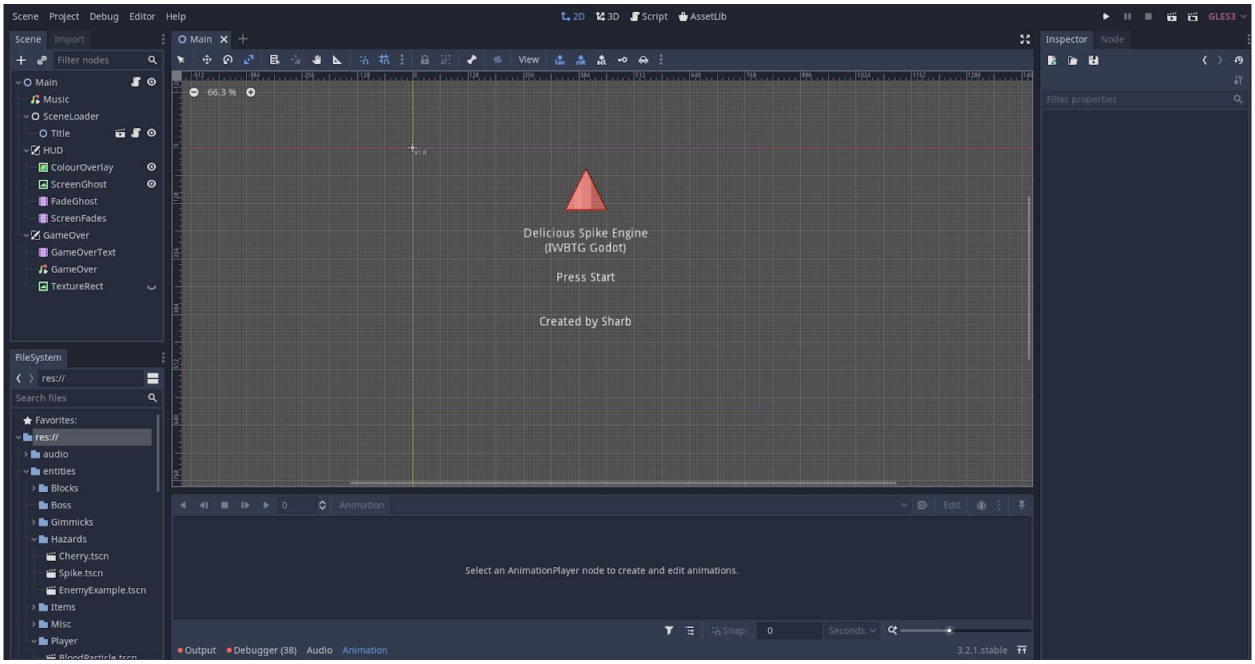
Godot

This document was made for version 1.0 of the Delicious Spike Engine made in godot v3.2.1

If you do not have godot you can access it from https://godotengine.org/

Main

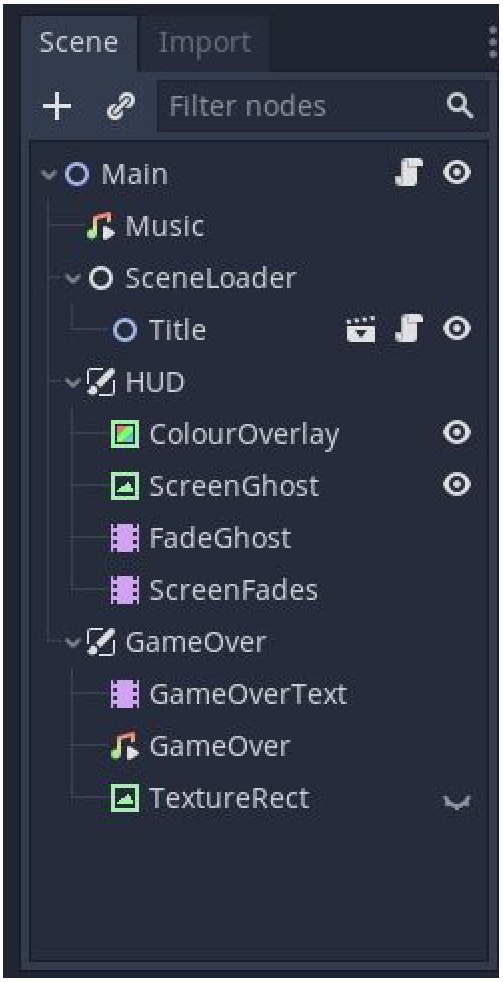
You'll find that when you open the project you'll be on the title screen



Select an AnimationPlayer node to create and edit animations.

This is in fact not the actual title screen but the global game window, or the equivalent of the "world" object that persists in most I wanna be the guy fangames.

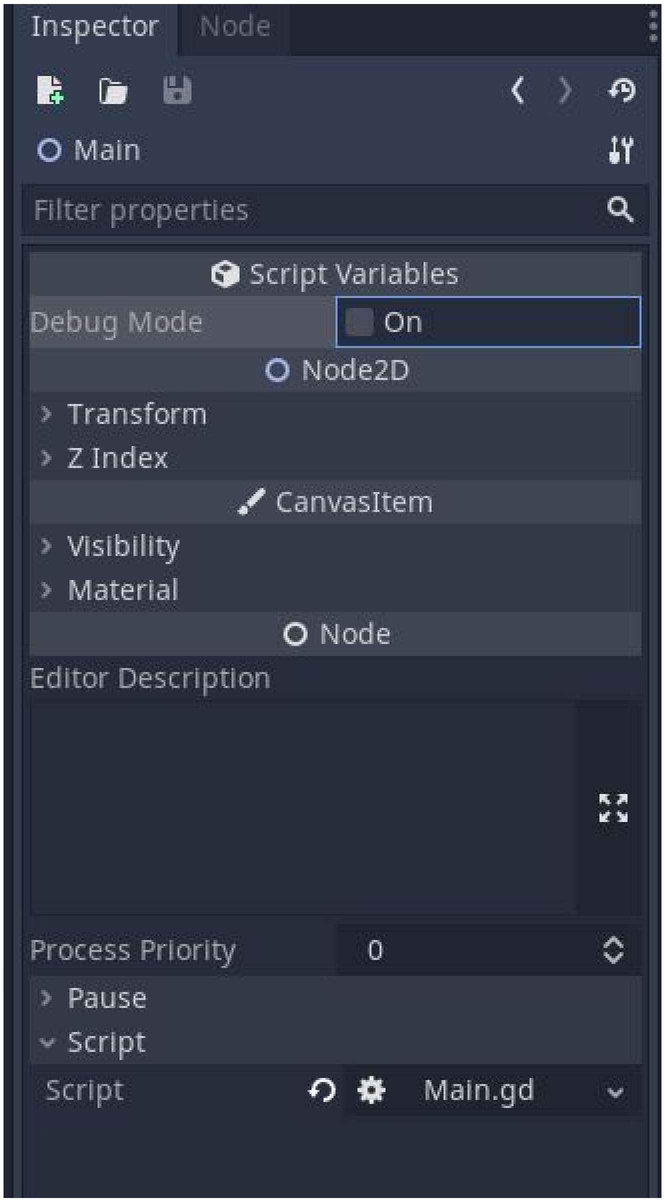
You’ll want to bring your attention to the scenes node list



Some of you might have noticed the node “SceneLoader”, there’s many ways to run your scenes in godot, my preference is to create a global scene and have a container for the scene I want to use.
This is what I’ve employed here and allows us to keep a consistency between scenes.

I’ll let you play around with the rest of the settings but the 2 nodes I want to cover here are SceneLoader and Main

Starting with main, I want you to select Main and then bring your attention to the inspector



Main

Filter properties

Script Variables

Debug Mode □ On

Node2D

› Transform

› Z Index

CanvasItem

› Visibility

› Material

Node

Editor Description

Process Priority 0

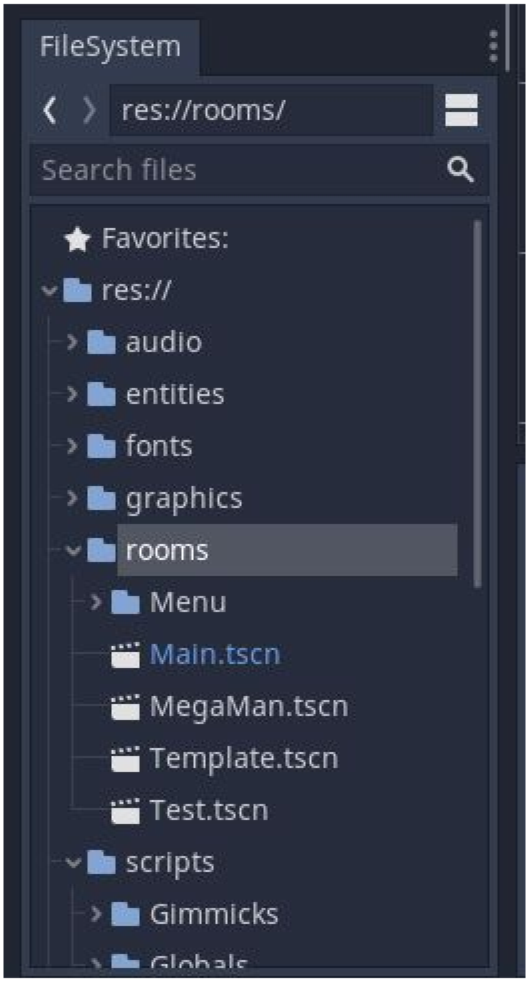
› Pause

⌄ Script

Script Main.gd

The inspector is important because it will allow you to easily manipulate your objects, but for main I want you to notice the “Debug Mode” option all the way at the top, I included this to help test your game, this will enable invincibility and allow you to click inside your game to teleport the player around, remember to turn this off when you export your project!

Next is the Scene loader, I wouldn’t recommend changing this from the title unless you know what you’re doing.
You should look in the file explorer and open the room folder

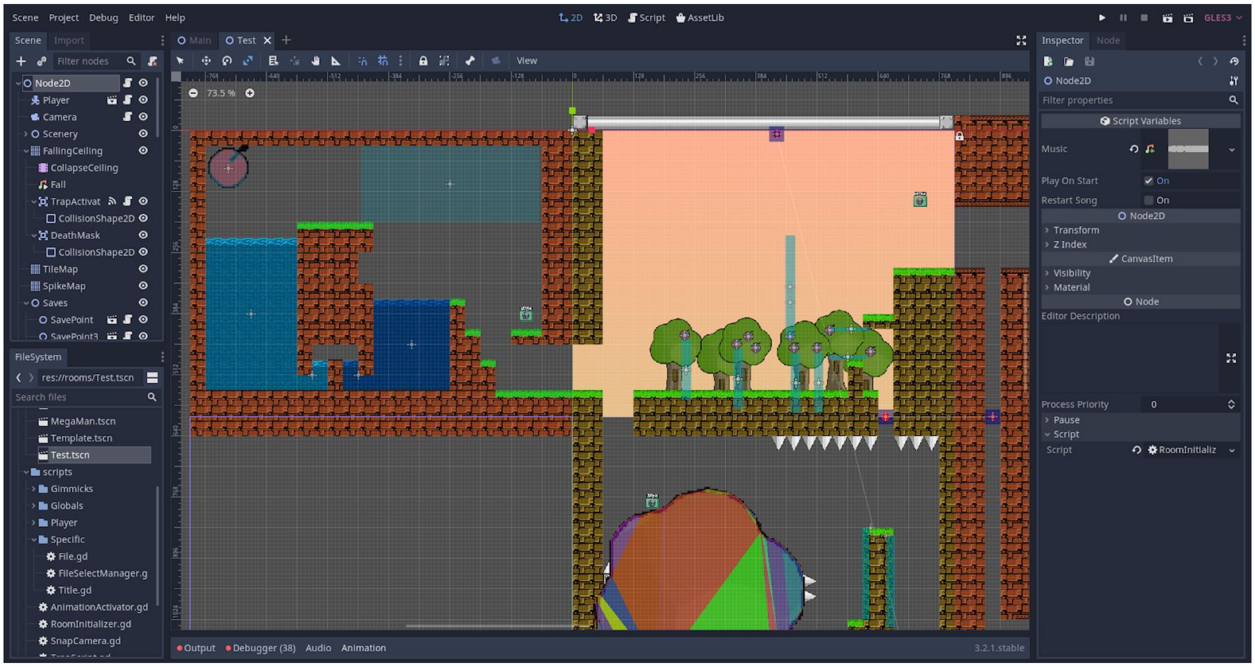

res://rooms/

These are going to be where you make your game rooms, if you wanna practice how scene loader works, delete the Title node, expand the Menu folder and drag “Title.tscn” onto the SceneLoader node in your scene tab.

You can also open these files and edit them by double clicking the scene you want to edit, for the next section I’ll be working in “Test.tscn”

Making a basic trap

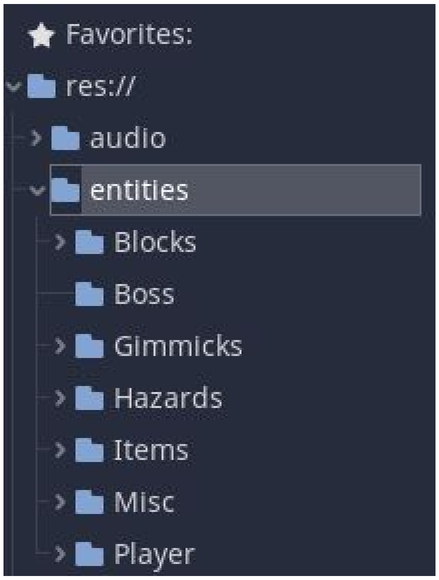
When you open test you should get a new tab that looks like this



Take some time to figure out how navigation works and what you can interact with, best way to learn how to use the engine is to play around with it, but for the next section we're going to build a basic trap.

I suggest also looking at the magnet icons at the top as those will toggle position snapping.

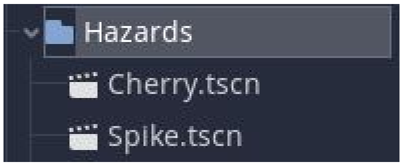
Browse the file system and look for the entities folder and expand it



Favorites:
- res://
  - audio
  - entities
    - Blocks
    - Boss
    - Gimmicks
    - Hazards
    - Items
    - Misc
    - Player

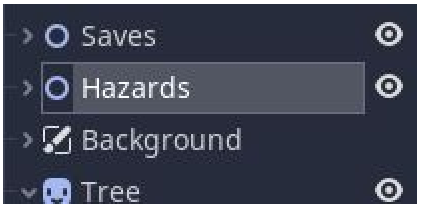
If you've used Game maker then the best way I could describe these are objects, while entities and rooms are the same type of object it's a good idea to seperate them so you don't get confused.

Now open up the hazards, and pick a favorite



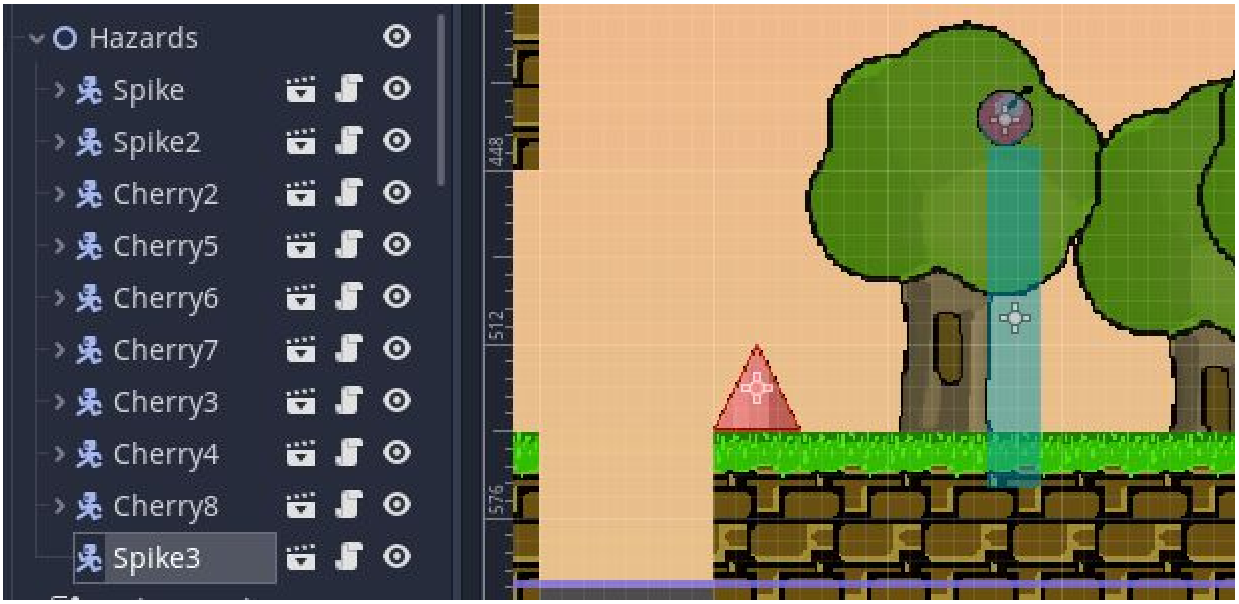
Hazards
Cherry.tscn
Spike.tscn

Too add it to the scene, first go to the **scene window** and select the Hazards node, while not necessary I think it's a good idea to keep them in a collection so you can keep scene navigation clean.



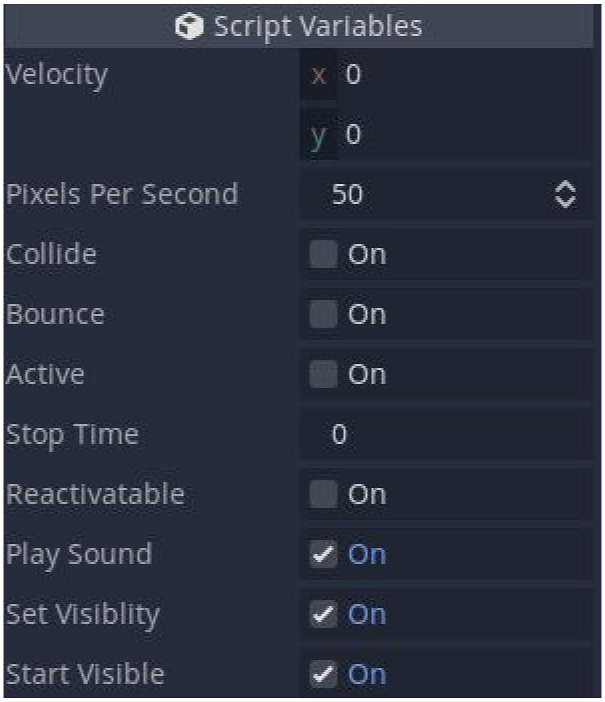
Saves
Hazards
Background
Tree

Go back to the hazard entities and drag either Cherry or Spike into the level editor screen, these two hazards are virtually identical beside theme, I'll be using spike to demonstrate, once you've dragged it into the game world



Hazards
Spike
Spike2
Cherry2
Cherry5
Cherry6
Cherry7
Cherry3
Cherry4
Cherry8
Spike3

Make sure your object is selected and go look into the inspector window and you'll see we have a large selection of options, I'll provide a quick rundown of what each does before making our trap activate

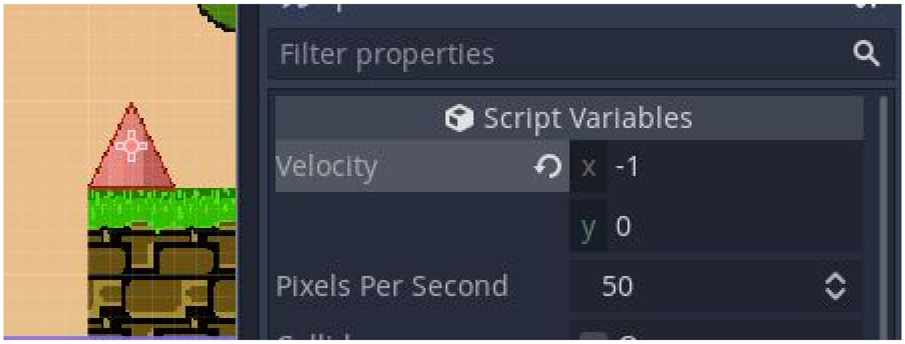


Script Variables

Velocity	x 0 y 0
Pixels Per Second	50
Collide	□ On
Bounce	□ On
Active	□ On
Stop Time	0
Reactivatable	□ On
Play Sound	☑ On
Set Visiblity	☑ On
Start Visible	☑ On

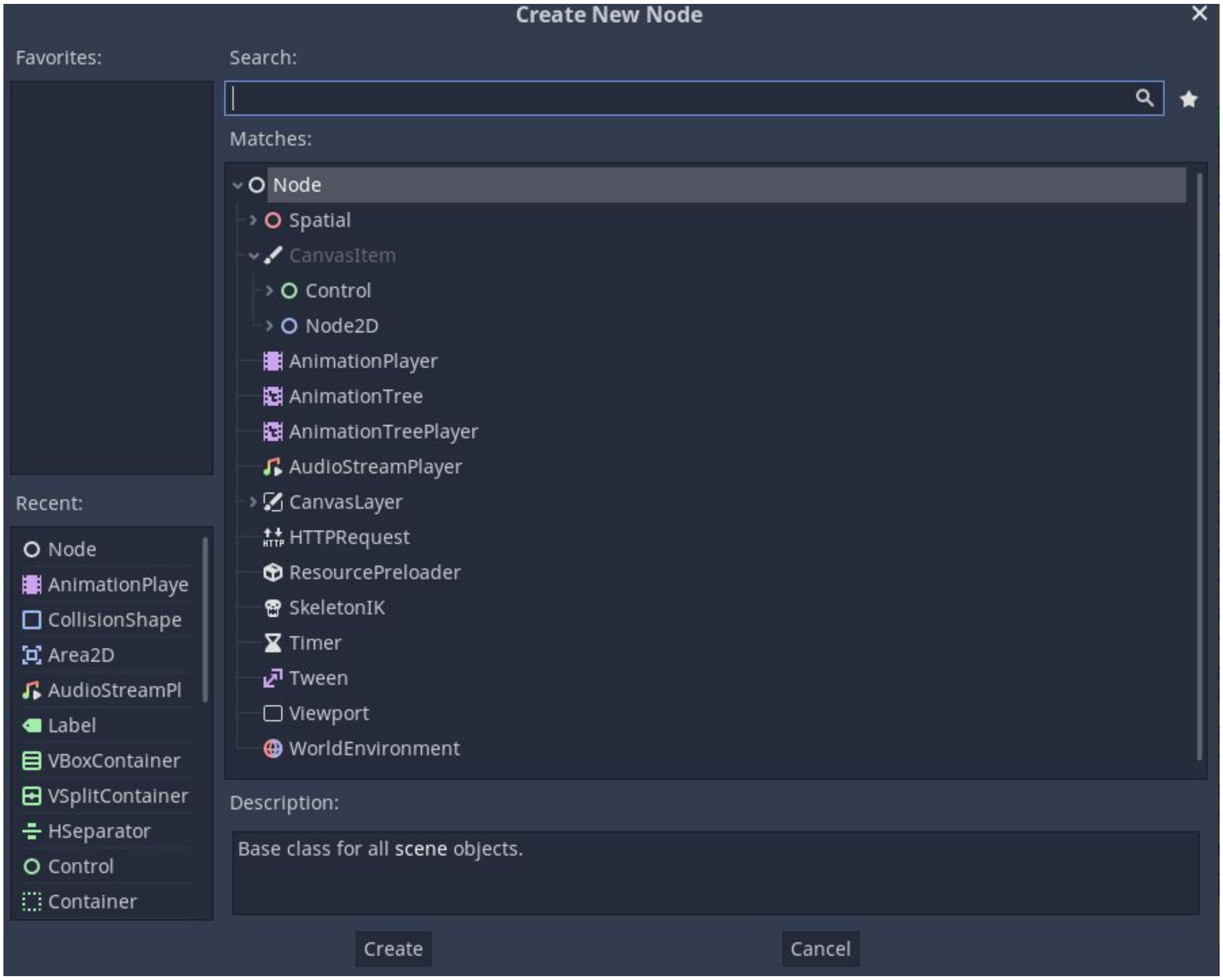
- Velocity - which way the object moves when activated, x 1 and y 2 would make the trap move down right with y making it move at double the speed
- Pixels Per Second (PPS) - how far the object will travel in a second (for example if the object is moving right by 1 then it will move 50 pixels right in one second)
- Collide - won't move if it encounters a collision mask that matches the collision layer the trap is on
- Bounce - if a collision is made it will take the collisions angle and bounce off of it (a flat surface will just move the object in the other direction)
- Active - activates the trap immediately without the need of a trigger (used for stuff such as moving platforms)
- Stop time - how many seconds before the trap deactivates (best to use this in tandem with pixels per second)
- Reactivate - works with stop time, if the trigger is activated again it will start the process over again
- Play sound - if a "LaunchSound" Audio node is found it will play that sound when activated
- Set visibility - will make the trap visible or invisible when activated (used for stuff like hidden blocks)
- Start visible - sets the visibility when the room starts, basically so you don't have to have the trap hidden in the editor

Ok now with that explanation over let’s set up our trap
Let’s make our trap move left when triggered, go to the velocity and set the x to -6



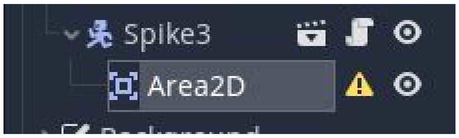
Velocity x -1
y 0
Pixels Per Second 50

Now you might be asking “how do I activate it”?
Well you’re going to need an area trigger!
Go back to the scene window and select the + icon at the top of the window, make sure your trap is selected.
a window like this should apear

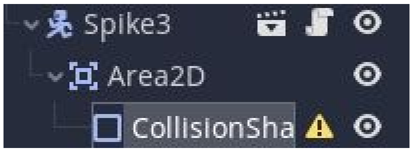


Base class for all scene objects.

We want to have an area2D node attached to our object, search Area2D, select it and then hit create, you should now have an area in your scene with a bos- exclamation symbol next to it

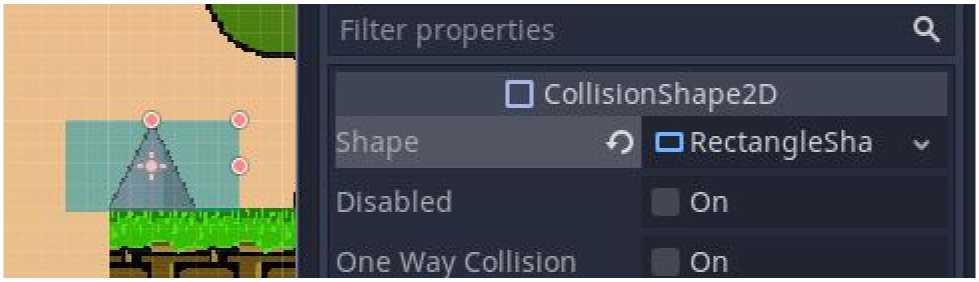


We need to create a collision shape for our area to use, so repeat the process you used to add the area but instead search for CollisionShape and add that to the area node (nodes under other nodes are considered children so I'll be using that term for future reference)



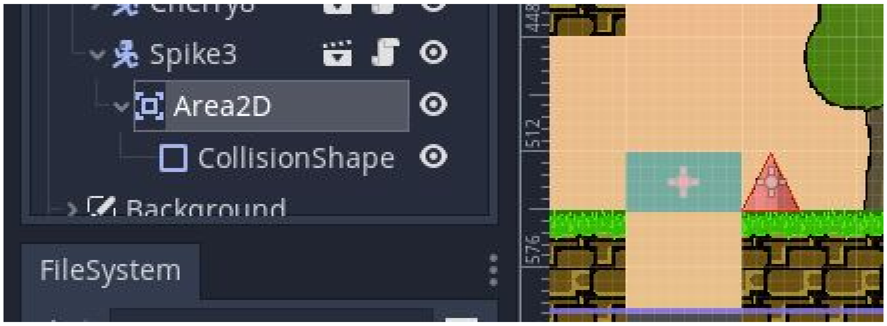
Now our collision shape is complaining it doesn't have a shape associated with it, go to the inspector, click the drop down arrow next to empty and add any shape you want, for this I'm going to use a rectangle, I'll let you play around with the collision shape so you can learn how the interface on those work since they're very straight forward.

Note: if you don't see your shape appear it may be because it was put at coordinates 0,0 press "F" to focus on where the node is

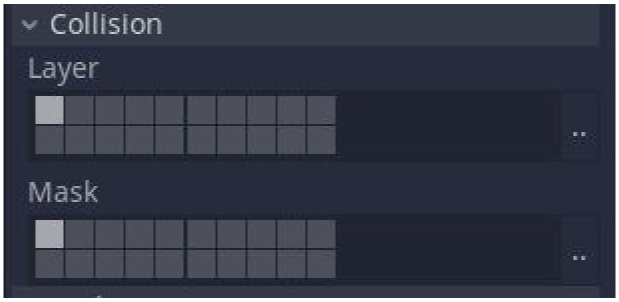


Shape	RectangleSha
Disabled	□ On
One Way Collision	□ On

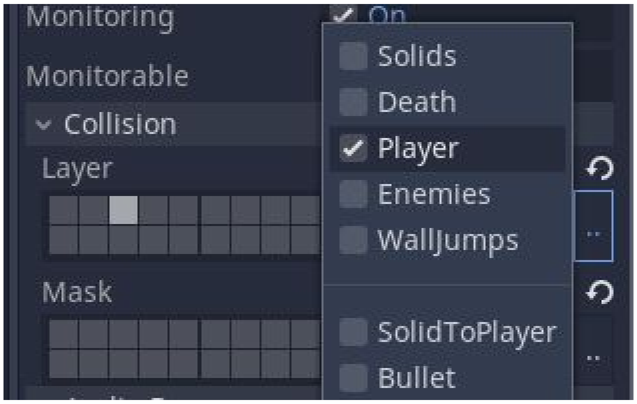
You may want to move the collision shape away from the trap, select the area2D node and move it to wherever you want to activate your trap



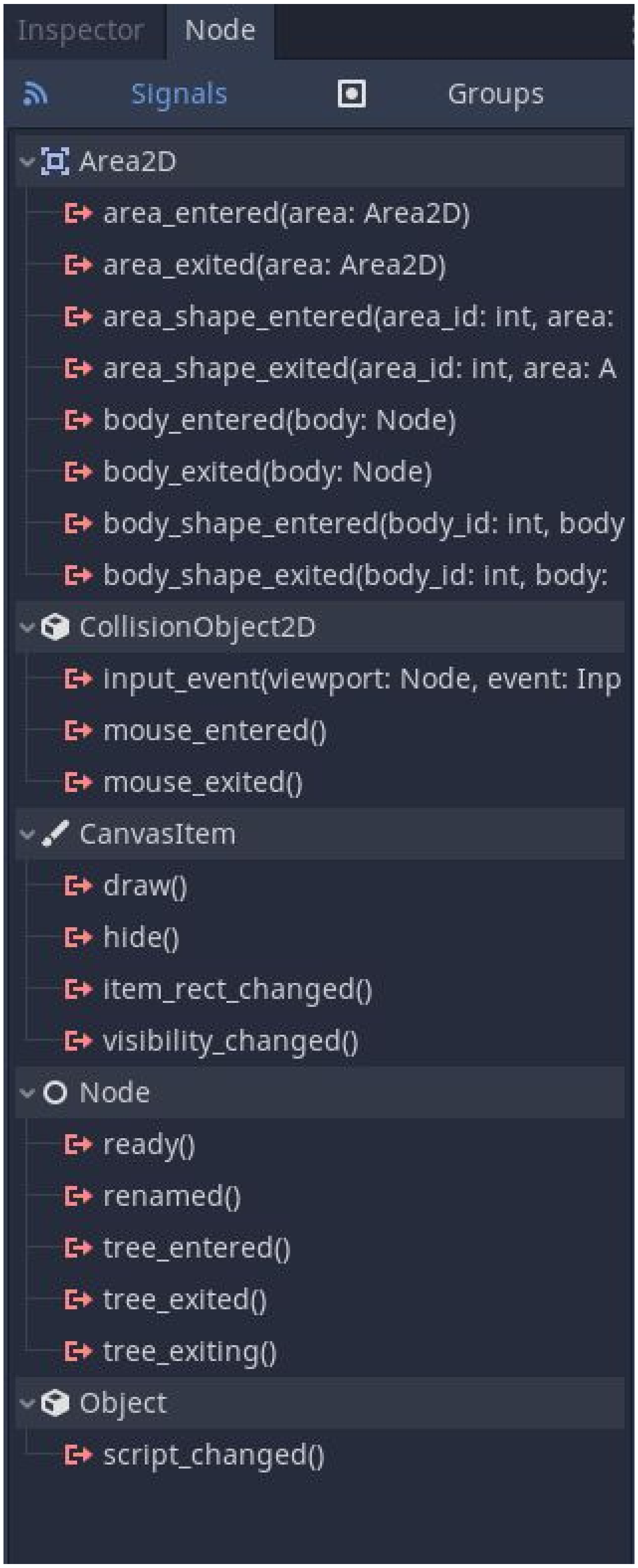
Now we want to make sure the player can actually trigger the area,
Go to the inspector and open the collision layers



Click the highlighted squares to disable and click the 2 dots next to the layer, check whatever you want to trigger the trap (yes, that means you can make almost anything trigger the trap, not just the player) for this tutorial check the player layer



We’re almost done making our trap activate, we just need to rename Area2D to “Trigger” you can do this by double clicking the name or selecting rename from the right click drop down menu,
Now we find the “node” tab and you should have a list of signals for the area



Inspector | Node

Signals | Groups

Area2D
- area_entered(area: Area2D)
- area_exited(area: Area2D)
- area_shape_entered(area_id: int, area:
- area_shape_exited(area_id: int, area: A
- body_entered(body: Node)
- body_exited(body: Node)
- body_shape_entered(body_id: int, body
- body_shape_exited(body_id: int, body:

CollisionObject2D
- input_event(viewport: Node, event: Inp
- mouse_entered()
- mouse_exited()

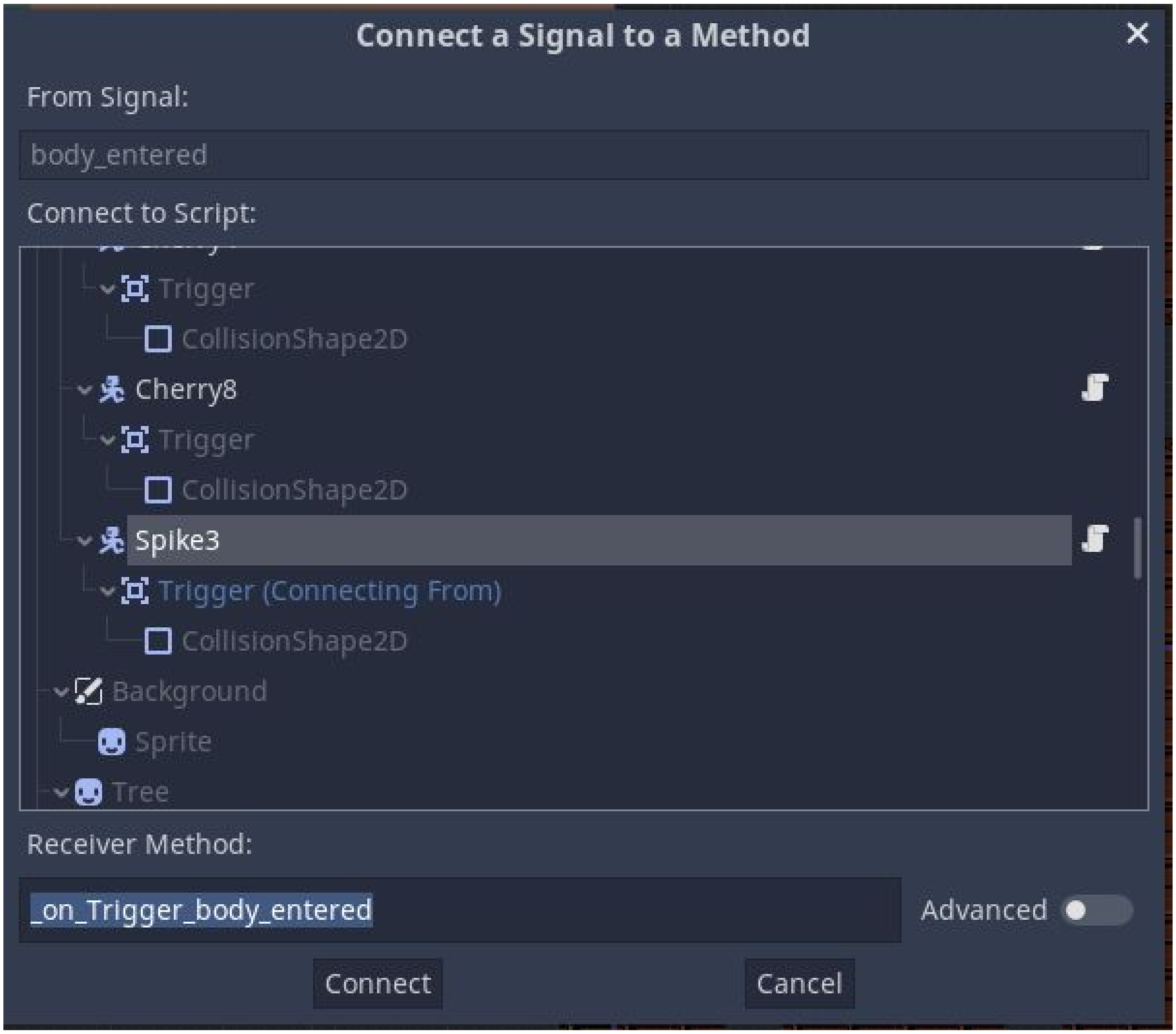
CanvasItem
- draw()
- hide()
- item_rect_changed()
- visibility_changed()

Node
- ready()
- renamed()
- tree_entered()
- tree_exited()
- tree_exiting()

Object
- script_changed()

Double click “body_entered” and you should see a list of nodes in your scene, find the node of your trap and select it.

Make sure the Receiver method says “_on_Trigger_body_entered”

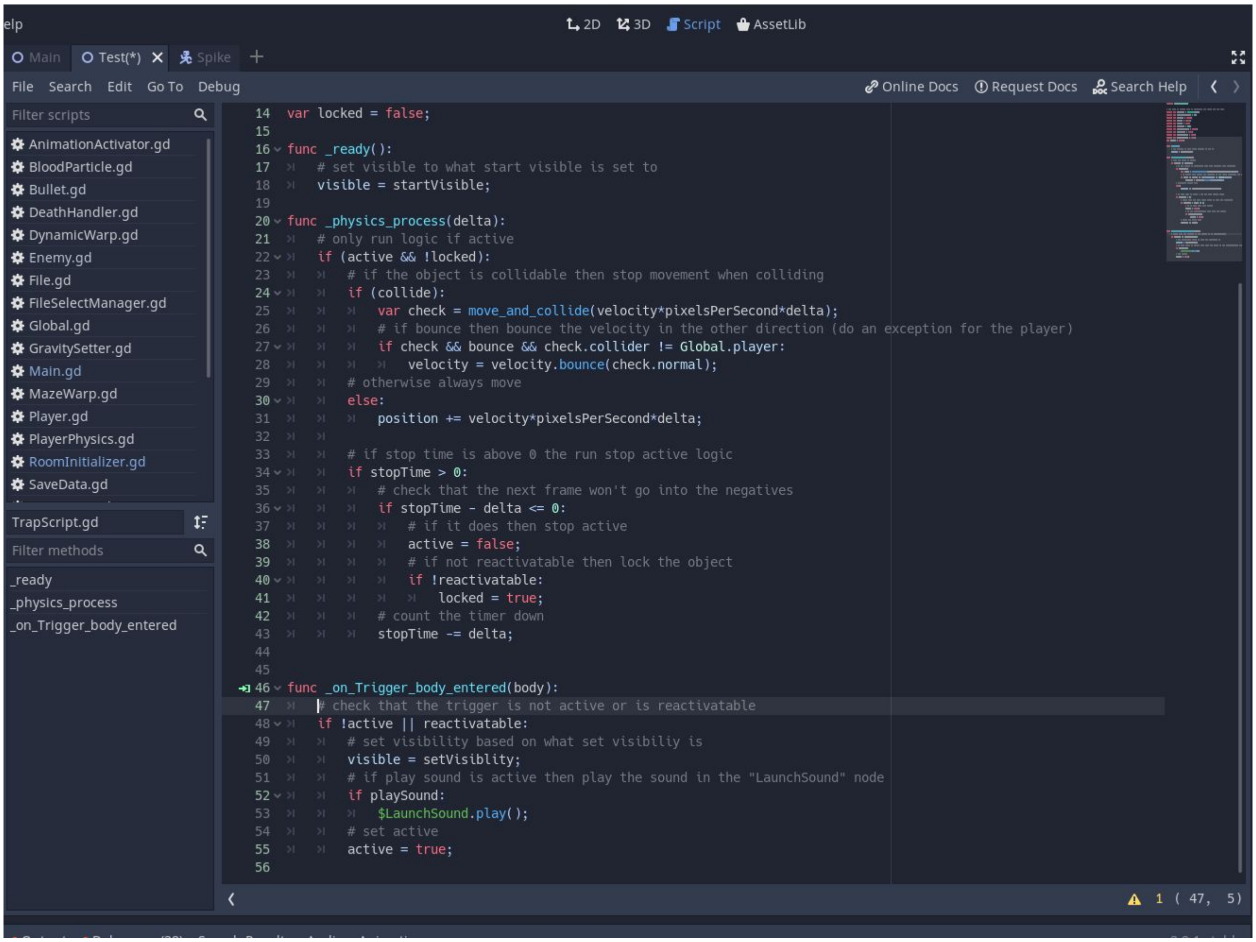


Connect a Signal to a Method

From Signal: body_entered

Receiver Method: _on_Trigger_body_entered

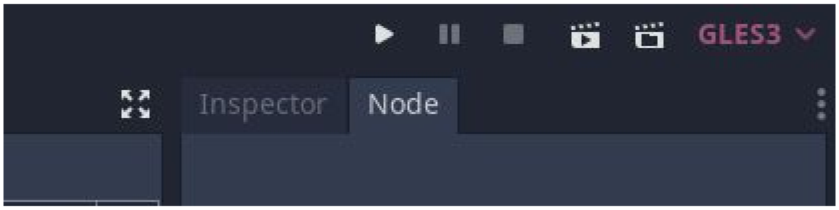
Click connect and you should be taken to the script menu, to go back to the editor go to the top of the editor and click 2D scene



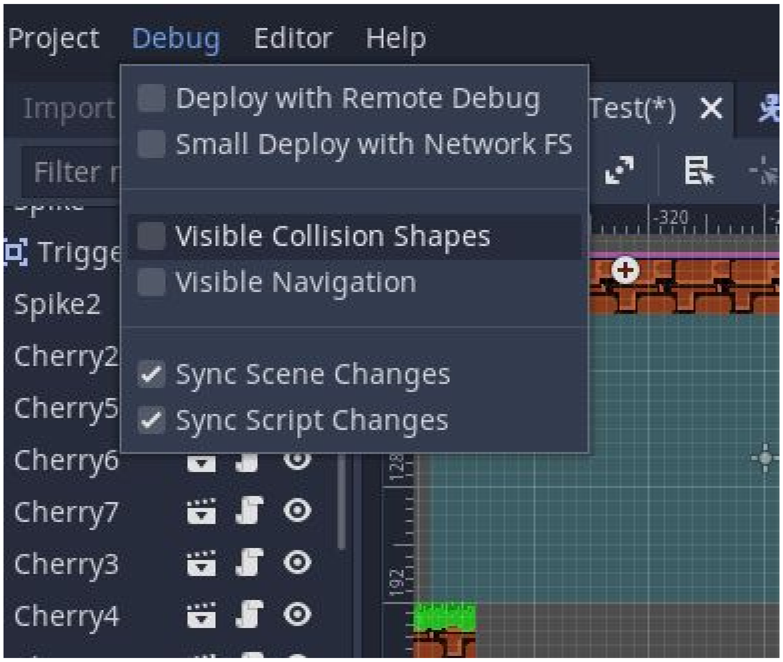
```
var locked = false;

func _ready():
    # set visible to what start visible is set to
    visible = startVisible;

func _physics_process(delta):
    # only run logic if active
    if (active && !locked):
        # if the object is collidable then stop movement when colliding
        if (collide):
            var check = move_and_collide(velocity*pixelsPerSecond*delta);
            # if bounce then bounce the velocity in the other direction (do an exception for the player)
            if check && bounce && check.collider != Global.player:
                velocity = velocity.bounce(check.normal);
        # otherwise always move
        else:
            position += velocity*pixelsPerSecond*delta;

        # if stop time is above 0 the run stop active logic
        if stopTime > 0:
            # check that the next frame won't go into the negatives
            if stopTime - delta <= 0:
                # if it does then stop active
                active = false;
                # if not reactivatable then lock the object
                if !reactivatable:
                    locked = true;
            # count the timer down
            stopTime -= delta;


func _on_Trigger_body_entered(body):
    # check that the trigger is not active or is reactivatable
    if !active || reactivatable:
        # set visibility based on what set visibiliy is
        visible = setVisiblity;
        # if play sound is active then play the sound in the "LaunchSound" node
        if playSound:
            $LaunchSound.play();
        # set active
        active = true;

```

Save your scene and you should be ready for testing!

To test click the play button in the top right of the window, then just play and find where your trap was placed and see if it works!



TIP: you can see collisions in game by enabling see collision shapes in debug

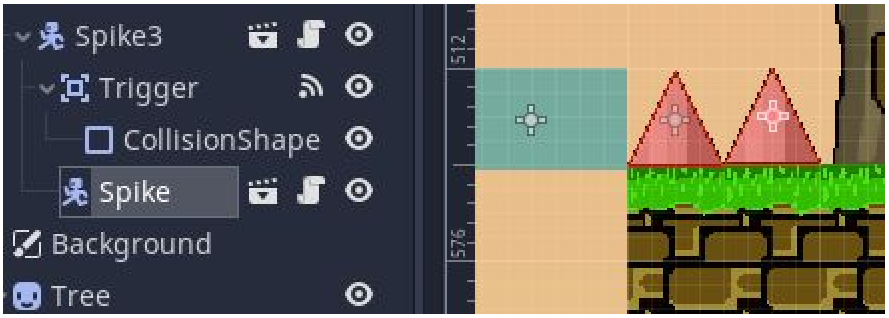


☐ Deploy with Remote Debug
☐ Small Deploy with Network FS
☐ Visible Collision Shapes
☐ Visible Navigation
☑ Sync Scene Changes
☑ Sync Script Changes

Are you a mad woman? Do you expect me to do this every time I want to set up a single trap!?

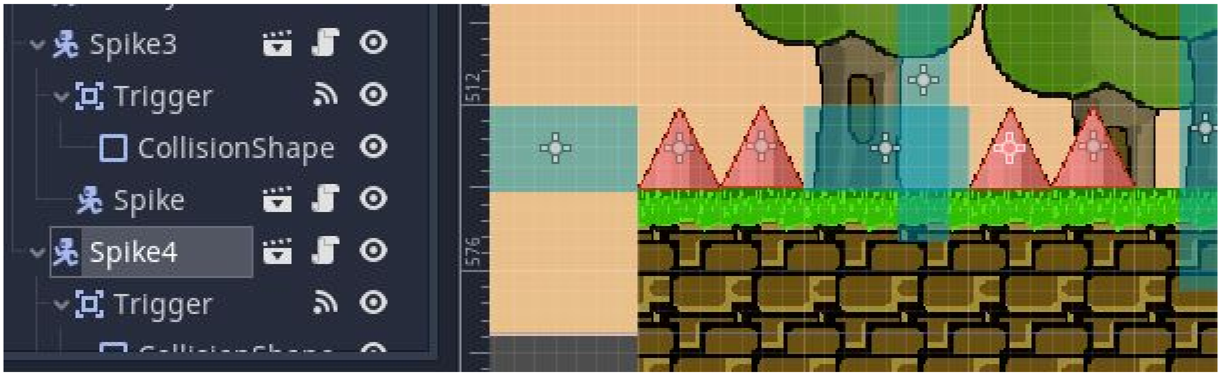
Nope, here’s two ways you can speed up the trap making process to make it 2 seconds

First is siblings,
Select your trap and drag a new trap into the editor window next to the other trap, make sure it’s a sibling of the first trap



When you activate the trap now, both the spikes move at once!!!
That’s because the children inherit their parents position, and if you’re the creative type, you may have realized you could have more then just hazards attached to the trap.
If you want an example, check out the megaman room, you’ll notice that not only are a lot of objects and blocks attached to the traps but blocks are also using the trap script for movement!

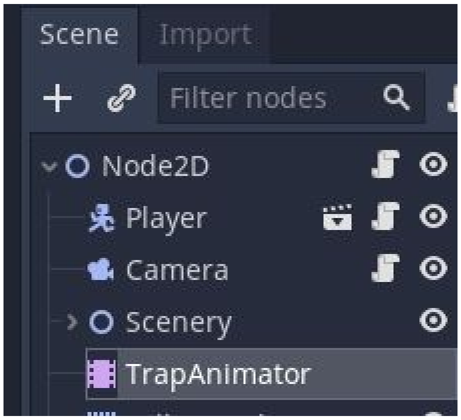
Now say you want another object to have it’s own trigger, easy, just select the original hazard in the scene window and press ctrl+D (or right click then click duplicate), move it to where you want and you’ll now have a duplicate trap with its own trigger. Be careful, your collision shape will be shared with the first so any changes you make to that will affect the first one, to fix that either click the shape drop down, click a new shape or make unique and it shouldn’t affect the first trap



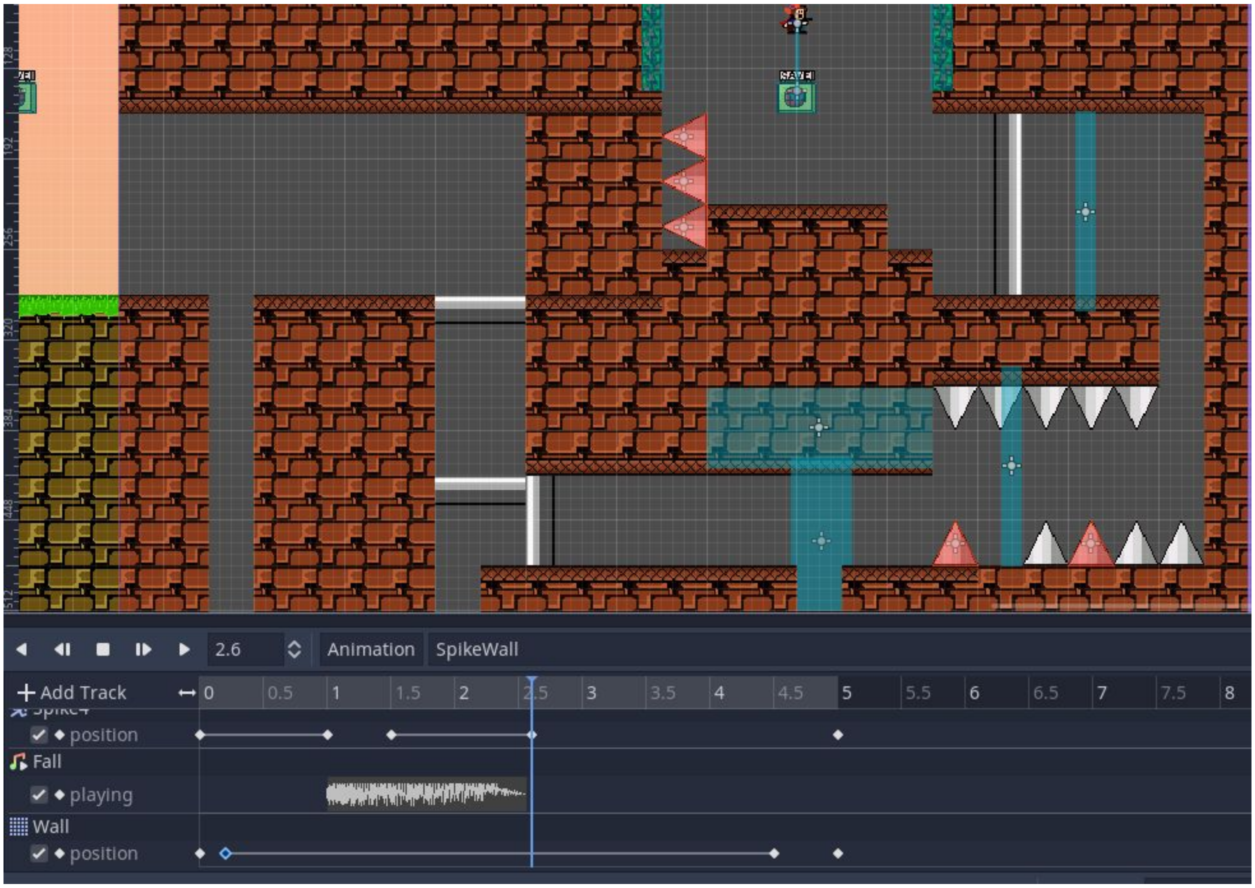
Animated traps

Basic traps are great but let’s say you want to have a bit more control of how your trap reacts to the environment? Like say, a spike wall trap similar to the first room in I wanna be the guy? That’s where the animation player comes in!

First find the TrapAnimator node (or add an animation player to the scene if you don’t have one) You’re going to want to get aquanted with how the animation player works which will probably take too much time to cover in this document so I suggest looking through the godot animation tutorial, try to make a wall of spikes that move in and kill the player, if you remember how we set up a trigger, you can do the same for a static body for a solid wall that can block the player in, or if you put the time in to figure out tile maps you can create a new tilemap for the wall to use.
Animation tutorial - https://docs.godotengine.org/en/3.2/tutorials/animation/introduction_2d.html

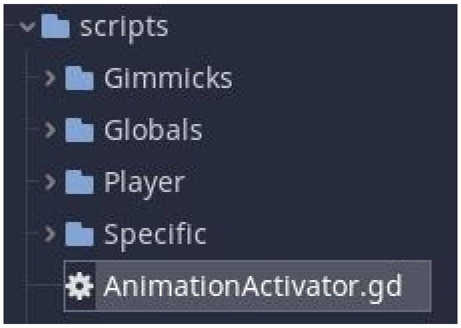


You’ll want to create a new animation for this next part, I’ll be calling mine “SpikeWall” and set it up anyway you want to play out

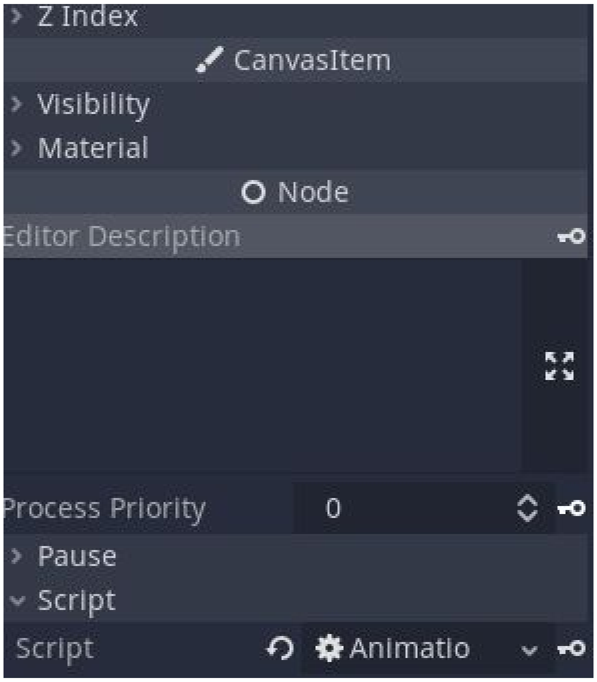


Now here comes the tricky part, repeat the steps for creating a trap trigger mask, don't worry about child or parents and come back to this when you get to the part you use the "body entered" signal.

Go to your scripts folder and find “AnimationActivator.gd”



Scroll to the bottom of the inspector and drag the script into the script box

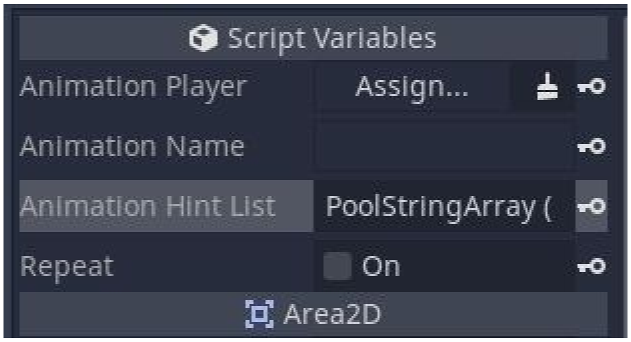


Process Priority 0

Script Animatio

Now go to your body_entered signal and make sure the receiver signal is called “_on_TrapActivator_body_entered” and then hook the signal back to the area trigger.

Go back to the inspector look at the script variables



Script Variables

Animation Player: Assign...

Animation Name:

Animation Hint List: PoolStringArray (

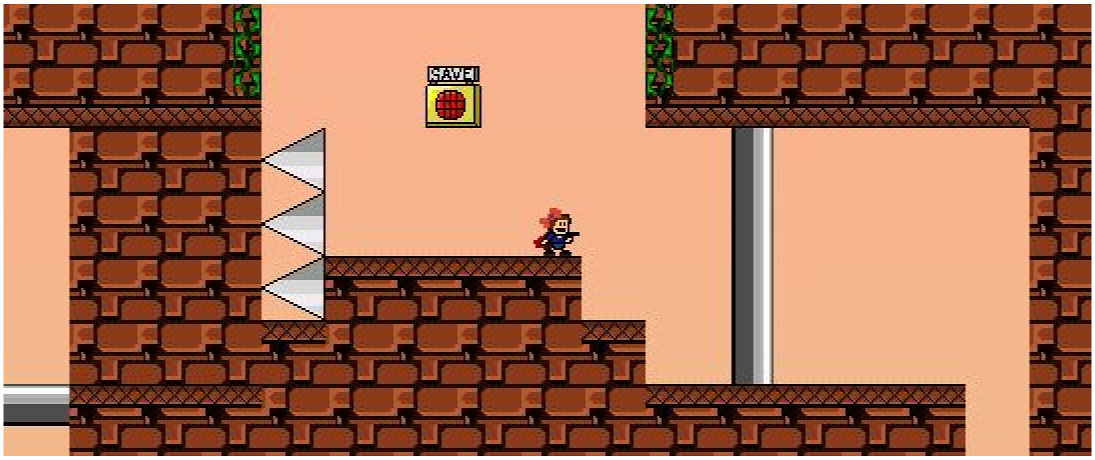
Repeat: □ On

Area2D

Assign the Animation player to the animation node we used to create our animation.
Set Animation Name to the same name as your animation, if you don’t want to retype or go to the rename menu to get the name, open the PoolStringArray next to the Animation Hint List, if you don’t see any animation names change the number and it should update the list. (I wanted to make a drop down but I couldn’t figure out how to update exported variables to show drop list downs)

Enable repeat if you want the trigger to activate the animator when triggered again, you may want to disable and reenable the mask collision in the animation if you don’t want the animation to queue up while it’s still playing.

If you set it up correctly it should play the animation when the player triggers it. If it’s not working make sure you followed the same steps for creating a trap trigger including the masking and layers



And that covers the basics for how to use the engine, in the next section I’ll give some brief notes on what you can do next.

Ending thoughts

Thanks for reading through this brief guide, hopefully it provided enough information for you to get an idea on how godot and this engine works.
If you're looking for what to do next I suggest learning Tile Maps and Tile sets
Tile tutorial: https://docs.godotengine.org/en/3.2/tutorials/2d/using_tilemaps.html
And just look through "Test" and "Megaman" and poke through the objects to see how these things work and interact.

As for I wanna be the guy.
If you want to learn more about I wanna be the guy fangames I suggest checking out https://cwpat.me/fangames-intro/

If you're a more experienced coder and aren't clicking with this engine, Smartkin has a much more compact I wanna be the guy engine that might interest you, you can find their git page here: https://github.com/Smartkin/godot-fangame-engine

As for further help, assuming it hasn't been 5 years since I've written this and discords disappeared off the face of the earth, I'm in the I Wanna Community discord server which is a server dedicated to all these games and helping people make them, go check it out (you can find the link in the fangames intro page I linked a few lines up)

Have fun!