Godot

Because games won't build themselves

{LAB ASTEK}

So, you want to a be a game developer? Well, if you are here, the question is quickly answered, welcome to a Godot workshop.

This workshop will guide you through making your first Godot project. You will learn how the Godot editor works, how to structure a project (LOL), and how to build a FUNCTIONAL 2D game.

Why 2D? 3D games are much more complex than 2D ones. You should stick to 2D until you have a good understanding of the game development process.

This can be quite useful https://docs.godotengine.org.

Set up the project

Open Godot and create a new project.

Now download this asset pack: https://kenney.nl/assets/platformer-pack-redux
We will need it later.

First you need to put your project in 2D. Then I must tell you a story: Godot works with a node tree system, where each node has only one parent (Unlike you I hope) and can have children.

But what's a node? Nodes are fundamental building blocks for creating a game. A node can perform a variety of specialized functions. However, any given node always has the following attributes:

- It has a name.
- It has editable properties.
- It can be added to other nodes as children.

If you need information about anything you can right click on a node to access the documentation of this kind of node.

So, you first need to create a 2D node before doing anything else, then launch your game by pressing F5.

Yeah… it doesn't work; you need to select a main scene which will be the one launched when you press F5. Let's see if you can find how to select the scene with your single node as the main scene.

When that's done launch your game, WOW what a great empty window, however it would be better if the window were a little bit bigger like let's say 1920x1080. So, go in the project settings and change that (you can also put it in Fullscreen if you are physically attractive).

Player scene

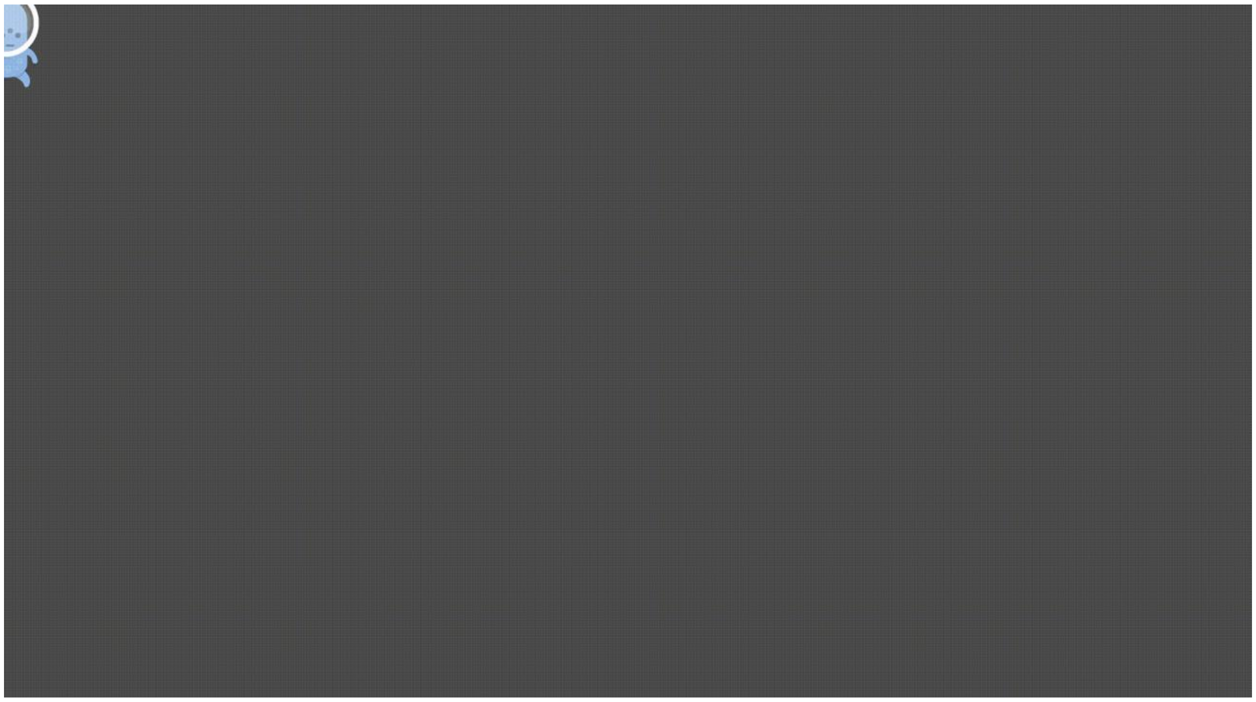
Now it's time to create a game because you know that's quite empty right now.

Create a new scene and change root node type to KinematicBody2D. Why a KinematicBody you ask, because it will allow you to add collision to the player.

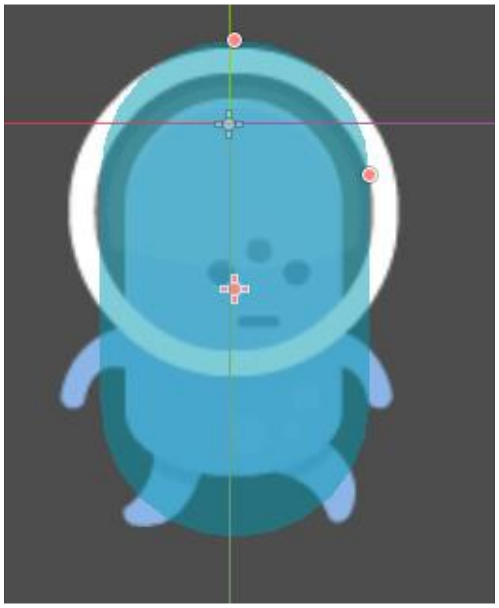
Now save your scene as "player.tscn".

Now our player need sprite, so add an AnimatedSprite node to the scene and create a new animation with the player sprite.

When that’s done press F6 and you should see your player in the top left corner.



Now we need to tell the engine which area of our player should have collision so for that we used CollisionShape2D. How your collision shape creates a new shape capsule where the player fit inside.



Now go back to your main scene and add an instance of the new player scene.

The map

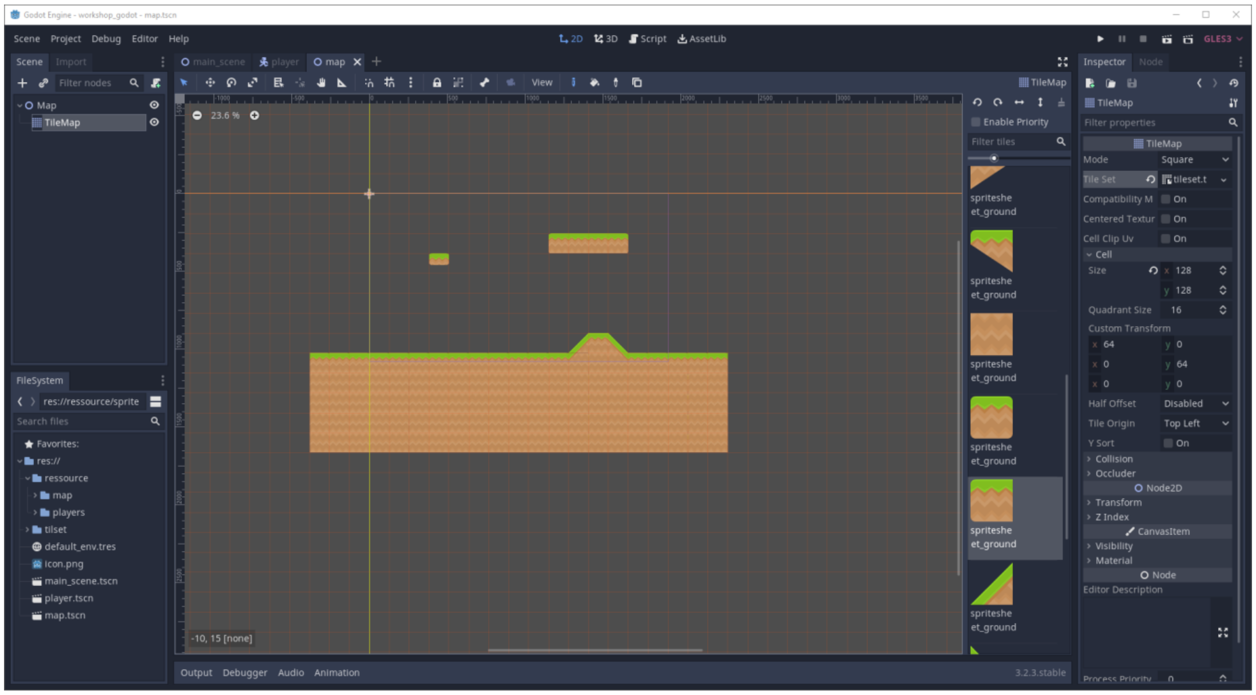
Now that you have a character with some animations, you might be thrilling with joy however, I am here to remind you that you just have a guy running in the void and that we are creating a platform game so adding some platforms is kind of mandatory.

The easiest way to do this is by using tilemaps. Create a new scene and choose a tilemap node to it.

Nice, however a tilemap is useless without a tileset so choose one (well create it) for your tilemap and choose the spritesheet grass for it.

With this you will be able to easily create some blocks that you will place to create your map. Add some tiles (Insane pro tip: activate the grid just under region), and since you are here, you might as well add collisions to the tiles.

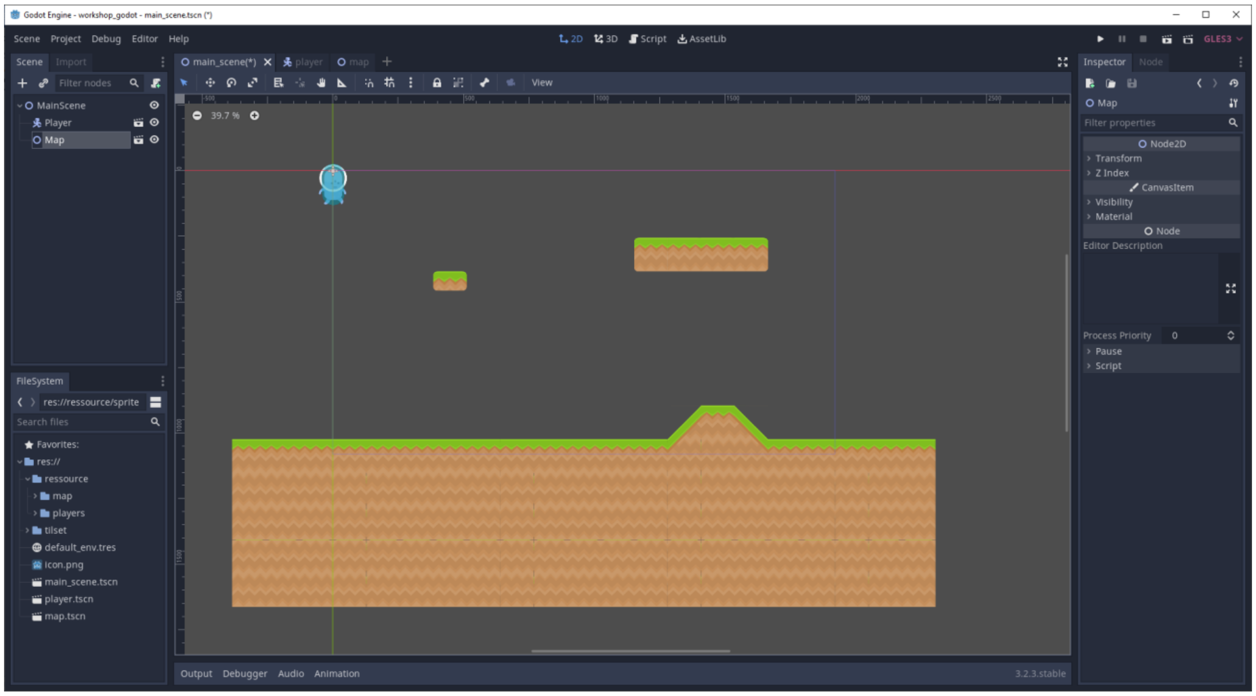
Now you can place some blocks! But wait a second... This is utterly disgusting the grid doesn't match with the size of the blocks. Change the cell size.



TileMap
Mode: Square
Tile Set: tileset.t
Compatibility M: On
Centered Textur: On
Cell Clip Uv: On
Cell
Size: x 128, y 128
Quadrant Size: 16
Custom Transform: x 64, y 0; x 0, y 64; x 0, y 0
Half Offset: Disabled
Tile Origin: Top Left
Y Sort: On

Finally go back to your main scene and add an instance of the new tilemap scene.

It should look like this:



Player movement

The player and the map are ready, however the player moves as much as you from your toilet after a spicy dinner.

First you need to assign the letters on your keyboard to some inputs. Go back to the project settings and DO IT!

Now we're going for the bad news, you need to code. I know, playing Minecraft and placing blocks is better, however all good things come to an end.

So, kids let's talk about scripts!

GDscript is the main language used in Godot (it's close to python) and even if you can use C#, C++ or VisualScript, I would advise you to stay with GDscript at least for this workshop.

Let us get this little guy in movement now. To this end you need to attach a script to your player KinematicBody2D

Wow it's filled with useless commentaries, you can remove everything, just keep the line: extends KinematicBody2D. It defines the type of node which the script inherits from (basically the type of the node where the script is attached).

Our first goal is to make the player move left and right. An extremely important function that you will use is _physics_process(delta) this function is call each time the engine calculates the physic.

Alright I will just give you some useful functions name and you will create a character that moves left and right (no gravity for the moment) with the correct animations (you probably will have to create new ones).

Input.is_action_pressed

play

flip_h

Also, I highly recommend using this variable: var motion = Vector2() and to use it with this function: move_and_slide, else your movement won't be pretty.

Also, calling other node is done by putting $name_of_the_node. In front of the function or variable from another node.

You might have noticed that the character can go offscreen, a node called camera2D can be quite useful to avoid this.

That's looking pretty good (I can't see it), but we still need to add two things: laws of physics linked to mass, and propelling power up; in other word gravity and jump. Since I'm lazy I'll do the same thing:

is_on_floor

Input.is_action_just_pressed

Hey, you're back! Now let's add your extreme remix heavy dubstep electro to the pile.

Sound effect and music

Now that our game looks good it also needs to sound good. So, for that we will add sound effect.

Adding a jump sound effect:

Open your player scene and add a new node of type AudioStreamPlayer. In the node's property select the jump sound file.

Now edit the player to play the sound when the player jumps.

Now guess how you add a music...

The main menu

Now that we have a game we need a menu.

So, for that create a new scene and set the root node type to Control.

Now you need to add a play button so the player can start the game. To detect when the button pressed is in your code you need to connect the pressed signal of the button to the root scene node.

Now edit your script to change current scene when the button is pressed.

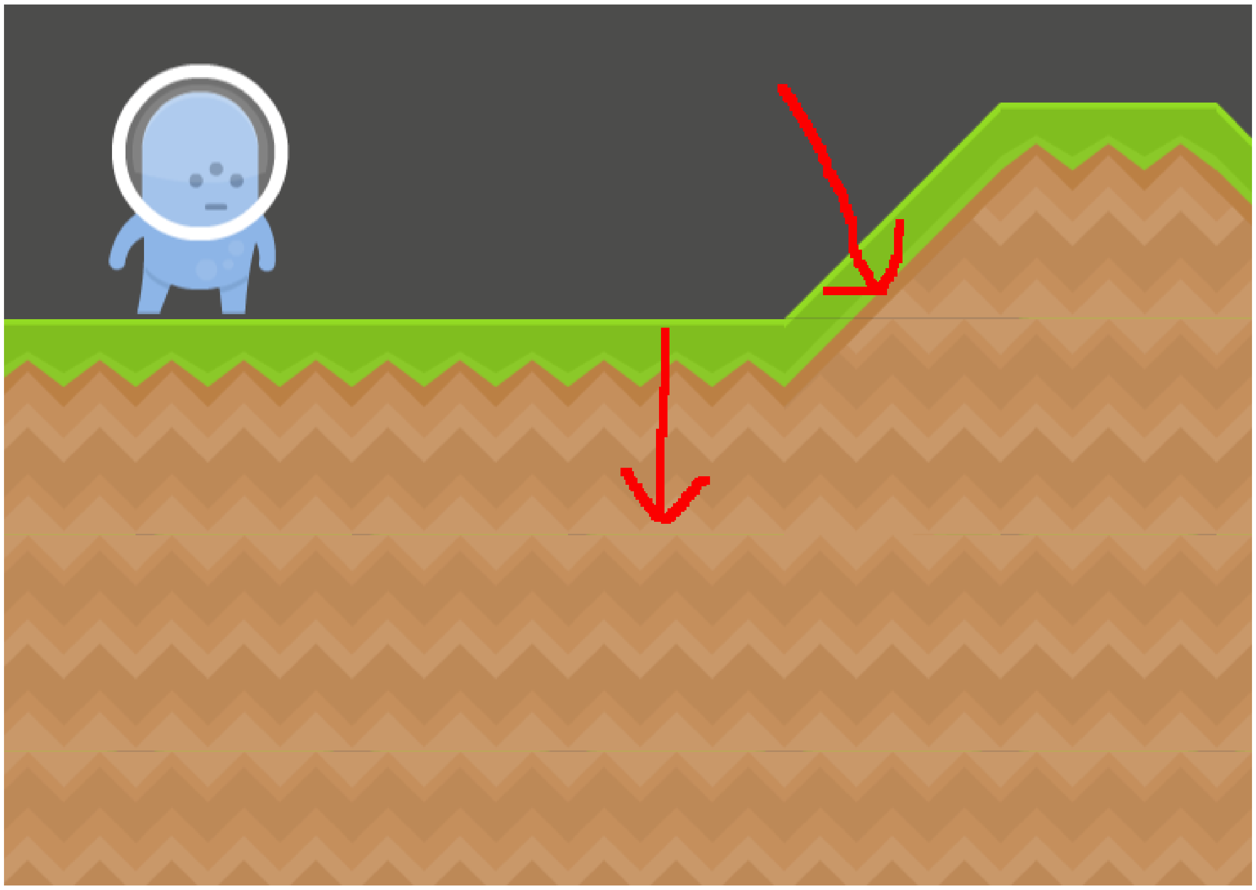
Run the scene and you should see your play button and it should start the game when you click on it.

If you want, you can add picture and label to customize your menu and make it unique.

Go back to the project settings and change the main scene to the game main menu.

Fixing the tilemap

If you've run your game, you've probably seen this line pattern on your tilemap



To fix that just go to the project setting and edit the rendering quality option, set the framebuffer allocation to 2d and enable pixel snap.

Export your project

Now that our game look awesome we need to export it so other person can play it!

So, for that open the export window and add a new export preset.

When that's done just click on export project at the bottom of the window.

And you're DONE!