Chapter Eight:

Tilemaps

In this tutorial we are going to explore the tilemap functionality built into the Godot game engine. A tile map is a 2D game map composed of layers of "tiles", which are essentially just a fixed size sprite with some additional properties. It allows you to quickly paint a level and reuse art assets, greatly decreasing storage and memory requirements for levels. In addition to supporting tile maps, Godot also includes a handy editor, although there certainly are some warts.

There is an HD video version of this tutorial available here or embedded below.

We are going to require two scenes for this example, one to create our tileset and one to actually use it.

Creating a Tileset

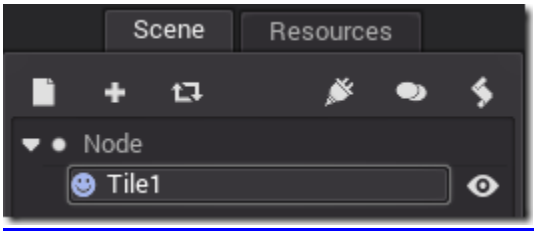
Before we continue though, I need a tileset to use. Instead of creating one, I am going to use this one from OpenGameArt.org. Of course, you can use whatever image you want, just be sure the tiles are the same size and have no spacing between them (if you want to keep the math easy).

Here is the tileset I chose:

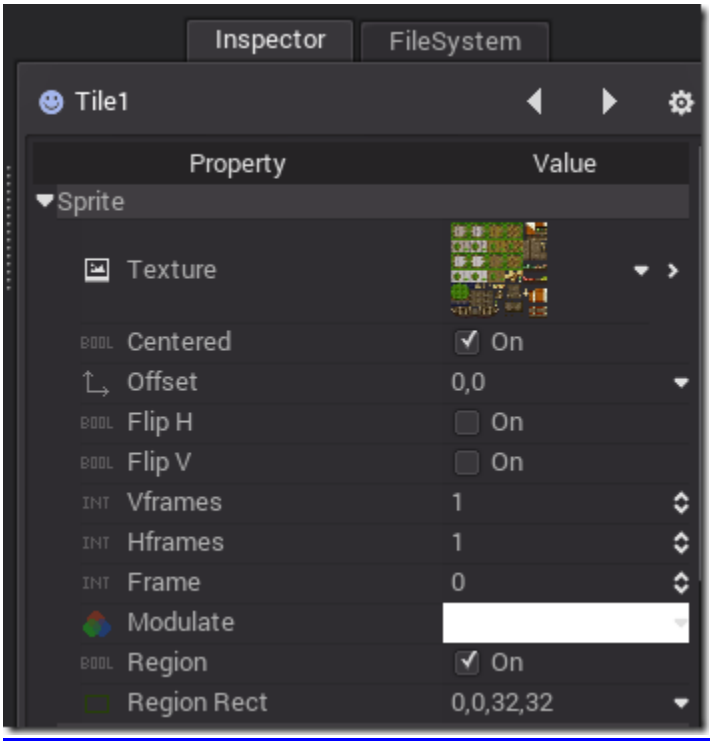
This version however is shrunk down, use the link above if you want to use the exact same tiles. Each tile in this image is 32x32 pixels in size. This will be important shortly.

Now let's create a tileset in Godot. Create a new scene. This process is really labor intensive unfortunately, but very powerful too. Create a root Node, then a Sprite for your first tile. Each tile needs to have a unique name. I am only going to use a subset of what is available on that spritesheet, but you simply repeat the process to add more.

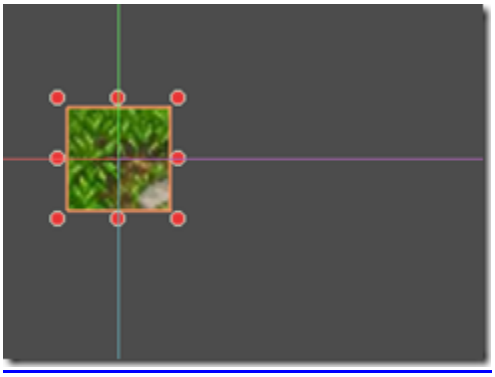
Create a hierarchy like this:



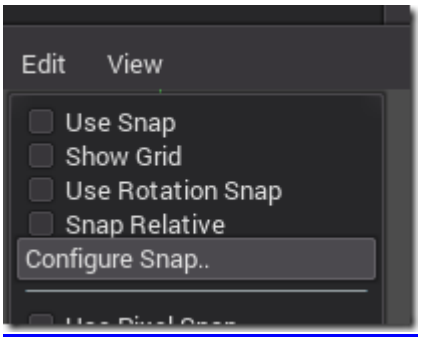
Now for Sprite Tile1. Now configure the sprite's Texture to your spritesheet. Importantly, set Region to on and set the Region Rect to 0,0,32,32, like so:



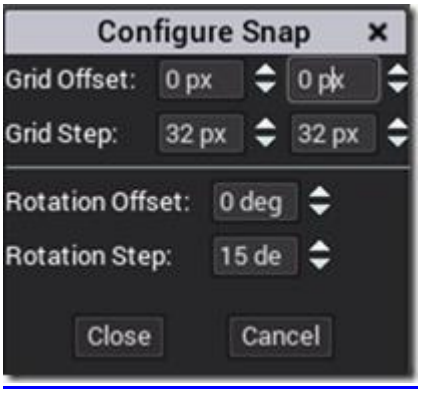
This sets the image source of this tile to a 32x32 region at the top left corner of the texture. Now you should see:



Now let's setup snapping, so we can arrange our tiles in a nice grid. Select Edit->Configure Snap

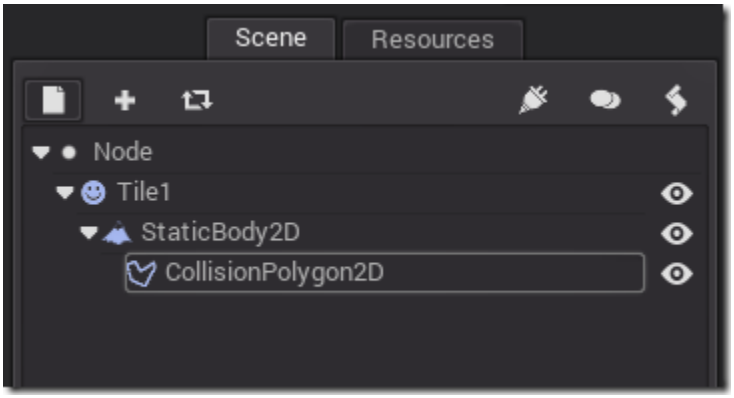


Set Grid step to the same dimensions as your tiles, in my case 32x32:

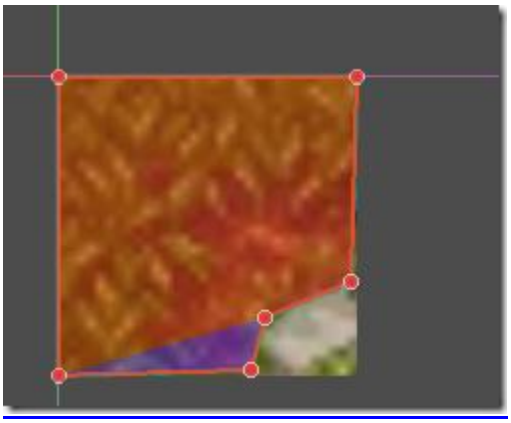


Now select "Use Snap" as seen in the menu above.

The next part is optional, if you want to use physics or collisions or not. If you do, we now need to set up StaticBody and hit detection boundaries. This process is the same as we went through in the previous tutorial so I am not going to go into details on how. You do want the end result like this however:

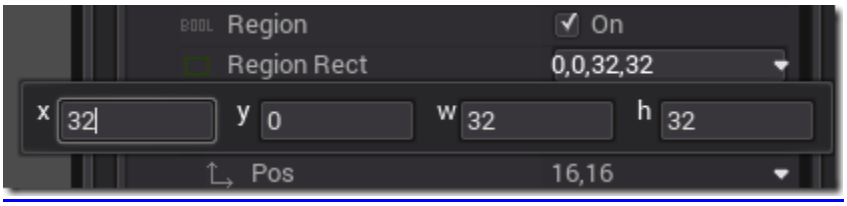


Then define the areas of your CollisionsPolygon2D that are collide-able or not, like so:



Turning snap on and off is critical here. Snap makes it easy to create the initial bounding box, as it will automatically snap to each corner. However, when you start adding fine detail, be sure to turn it off.

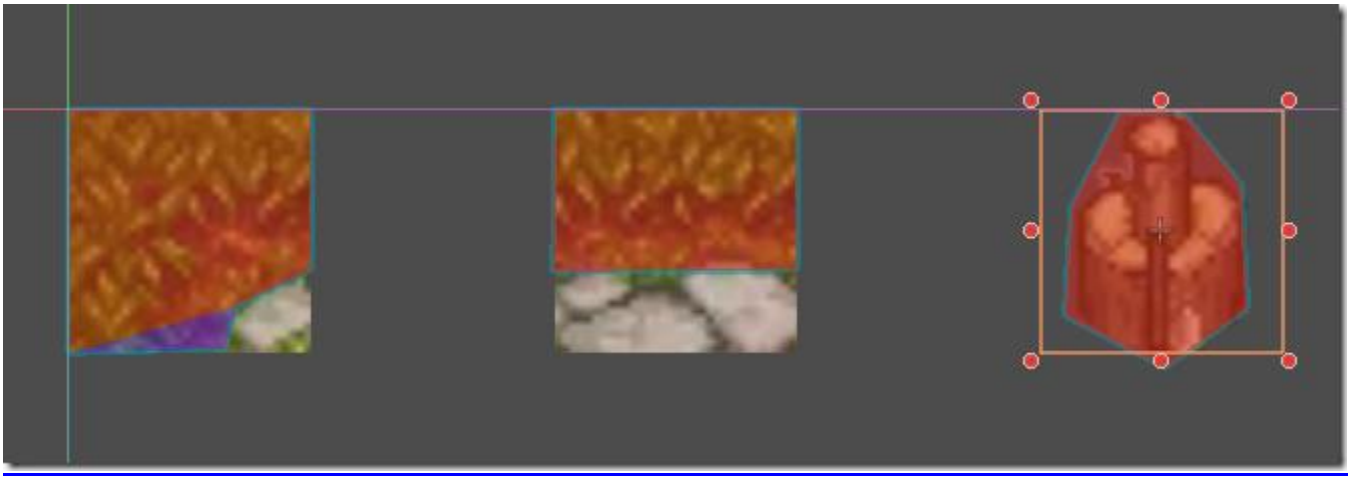
Now repeat this process for each tile you wish to include from your spritesheet. You can make the process a great deal quicker using the Duplicate option (or Ctrl +D). Simply select your first tile then duplicate. Then you just need to reposition it on the map, redefine the CollisionPolygon2D bounds and of course, update the region to the next tile, like so:



This will select the next tile from the top left of the image. Repeat until you have all of your tiles defined. Yes, it would be nice if this was an automated process!

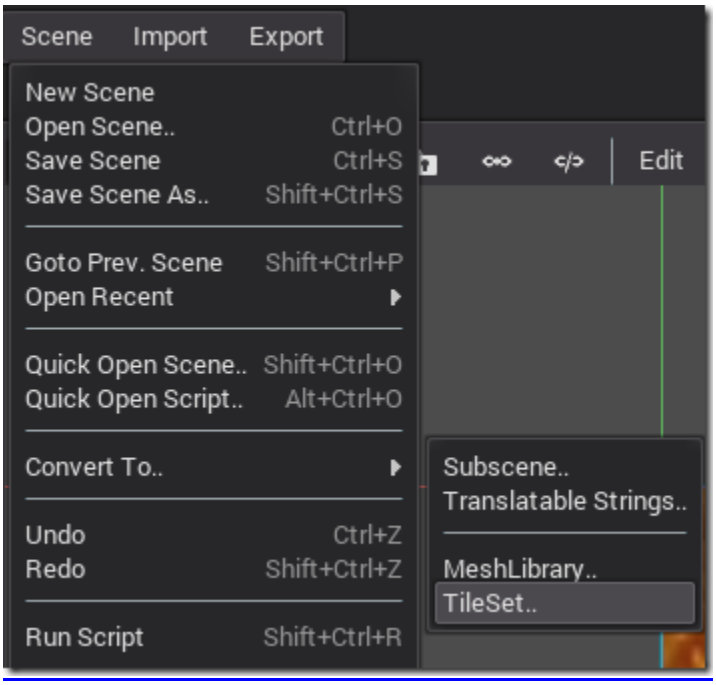
Because I'm lazy, im only going to define three tiles, like so:

Which look like this with collision polygons defined:



Please note, there was no need to put space between each tile like I have here. Now that we've defined our tiles, first save your scene, I called mine tilemap.scn.

Now we are going to export a tilemap. Simply choose Scene->Convert To...->TileSet...



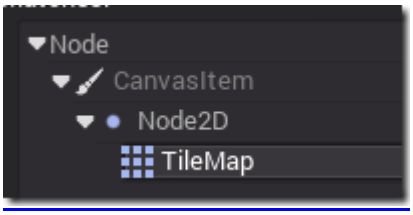
I called mine tileset.xml. Be aware of the option Merge With Existing. When you make changes to your tileset in the future, you probably want to turn this off if you want your changes to be completely overwritten.

I personally found the Collision polygon exporting to be extremely buggy. You may want to open up the generated XML file and make sure collision information was properly exported:

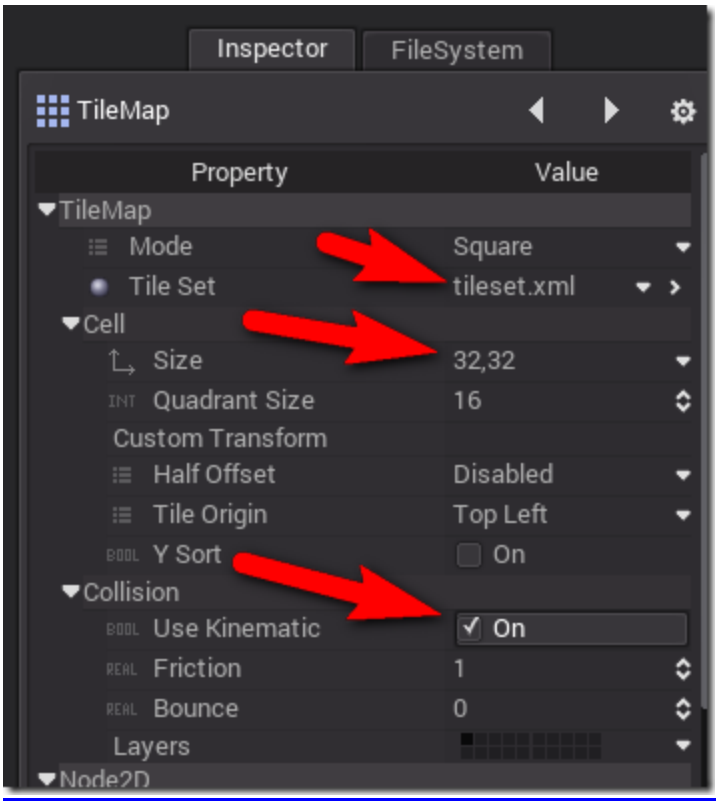
Using a Tilemap

Now that we've got our tileset created, our collision polygons configured and all things ready to go, it's time to create a new scene then create a Tilemap.

The Node you want to add is a TileMap, like so:

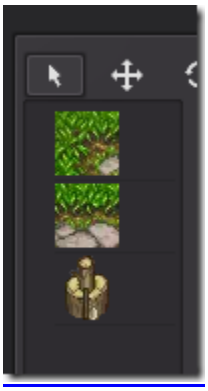


Configure your tilemap like so:



If you are using Physics, set Use Kinematic on, otherwise don't. Set the Cell Size to the size of your tiles and tileset to your newly exported tileset.

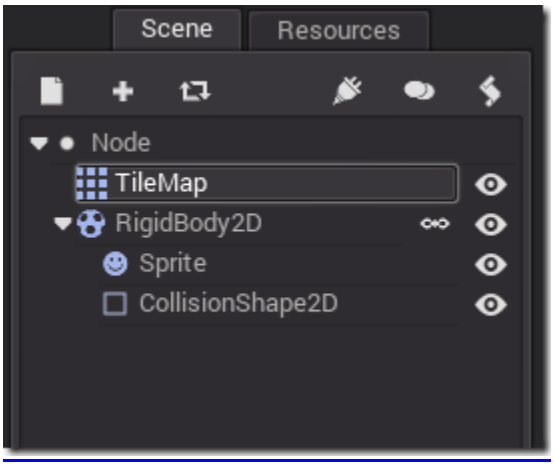
You will notice now, with the TileMap node selected, your tiles will appear down the left hand side of the 2D editor window:



You can now use these to "paint" your level, like so:

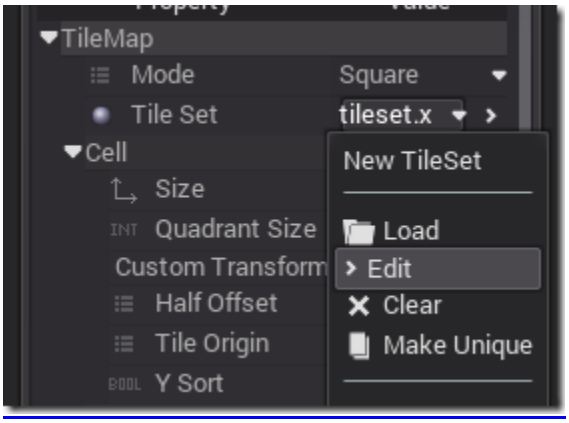
You can create multiple TileMap objects if you need to have different layers of tiles (like foreground props for example). Today though, we are going to stick to this simple example.

Finally I created a RigidBody sprite to interact with our tilemap, so our final scene looks like this:

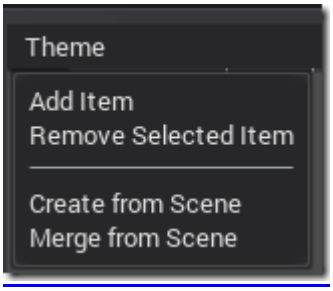


Now when we run it, it should look like:

You can make some edits to your tilemap from within Godot without re-exporting the tileset. With your Tilemap selected, select your Tileset property, then Edit:



You will notice a new Theme menu appears, allowing you to edit your Tileset, such as adding new items:



You can also see the tiles that make up the tileset in the Inspector: