Creating terrain sets (autotiling)

Note

Godot 4's terrains are a more powerful replacement for Godot 3.x's autotiles. You could only set autotile bits either on or off. But, with terrain tiles, you can set each bit to a specific terrain. When you paint terrains on the TileMap, Godot chooses tiles to handle transitions between terrains automatically.

In addition, an autotile in Godot 3.x was a specific kind of tile, different from an atlas tile. But, in Godot 4, a terrain tile is still an atlas tile, but with terrain data assigned. This means that you can use it as a terrain or as a single atlas tile. And any property that you can assign to an atlas tile, you can also assign to a terrain tile.

When you create a game level using tiles, you may need many variations on a single tile with different corners or edges. You might use them to create complex shapes in a platformer, to draw walls or cliffs in a top-down level, or to transition between different terrains on a world map.

You can end up with a large number of tile variants, and it can be tedious to place them all manually. Using them for procedurally generated levels can be difficult, too, and require a lot of code.

Godot offers *terrains* to automatically handle the placement of these kinds of tiles.

Terrains can be complicated to understand. This section starts with introductions to how terrains and terrain modes work. If you're ready to create your own, go to Setting up a new terrain set. If you're an experienced user or you're here to troubleshoot, you may want Special cases.

Hint

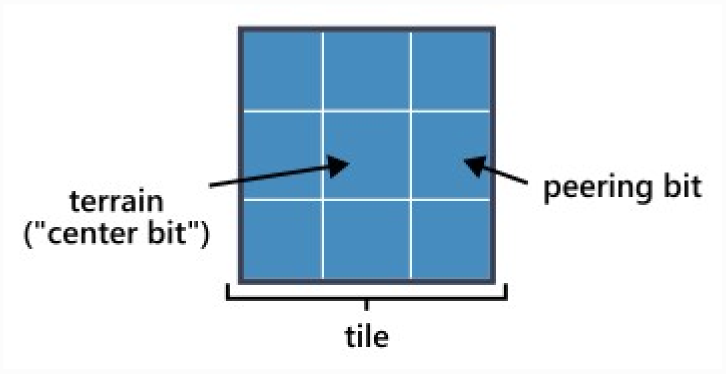
All the examples in this section use public domain tilesheets. To use any of them in your own project, right-click and save the tilesheet's image to your project folder.

You can also download the starter project 2d_using_tilesets_terrain_starter.zip. It contains scenes for every example with the TileSets and terrains already set up.

Understanding how terrains work

The terrains system is made up of **terrain sets**, **terrains**, and **tiles**. A TileSet can have one or more *terrain sets*, a terrain set can have one or more *terrains*, and a terrain can have one or more *tiles*.

Terrain tiles have one *center bit* and multiple *peering bits*. Each bit can have one *terrain*. We often refer to a tile's specific configuration of terrains assigned to center and peering bits as its **bitmask**.



A tile with center and peering bits in Match Corners and Sides mode.

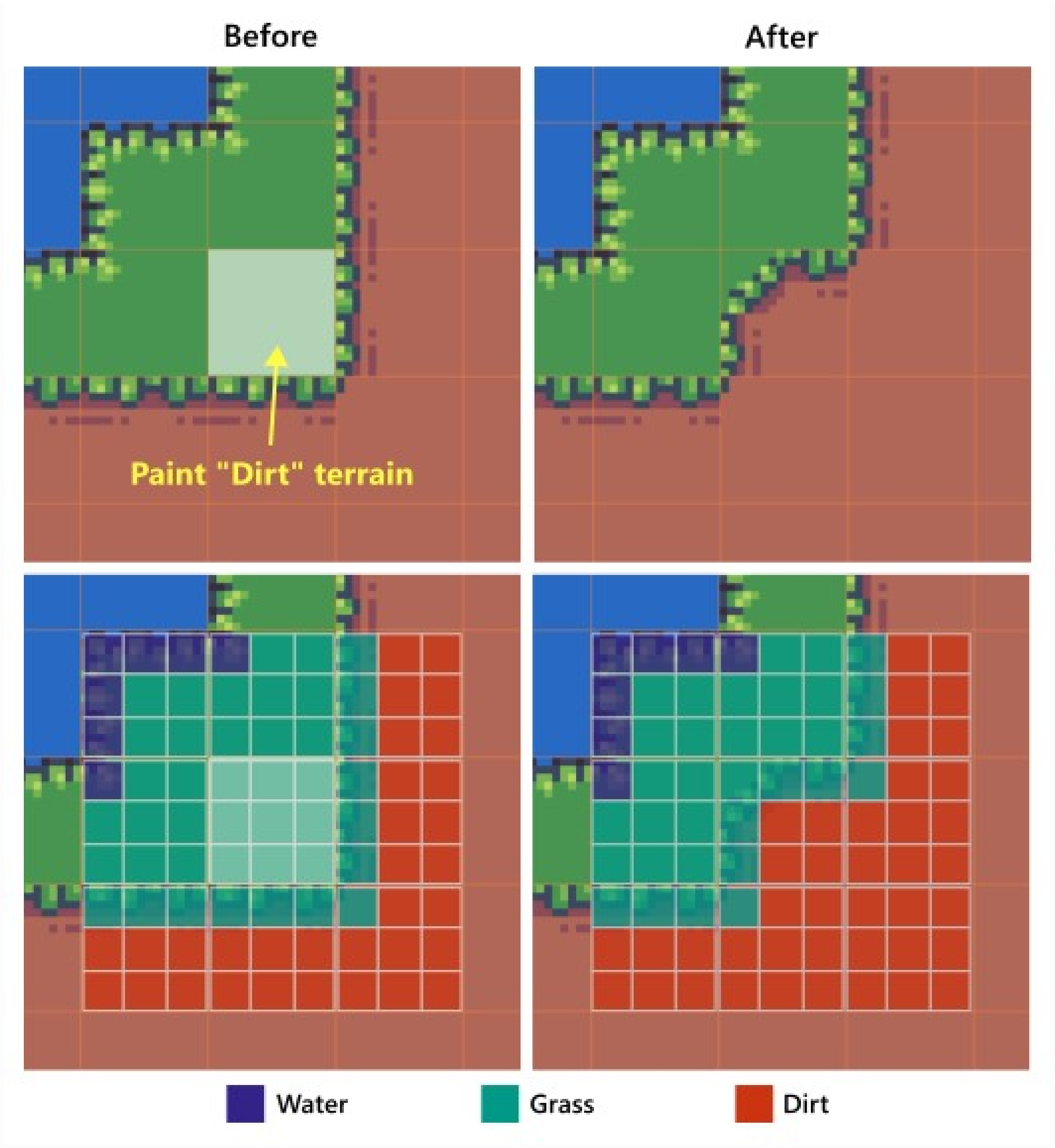
The center bit, or the tile's **terrain**, is the terrain the whole tile belongs to, while **peering bits** are like the edges of a jigsaw puzzle piece: they determine which other tiles they can fit next to. The tile's shape and the terrain mode determine which peering bits it has, and what they look like.

In this tilesheet, there are three terrains: water, grass and dirt:

Adapted from The Field of Floating Islands by Buch (CC0).

You would assign the water tile to a `Water` terrain, the grass tiles to a `Grass` terrain, and the dirt tiles to a `Dirt` terrain. Then, instead of placing the individual *tiles* onto a TileMap, you could paint individual *terrains*, and Godot would automatically choose which tile to put in each cell.

As seen below, when you paint the `Dirt` terrain onto a cell, Godot chooses a tile that has the `Dirt` terrain as its *terrain* (center bit), and whose peering bits match the neighboring tiles.



Before and after painting in Connect Mode.

In *Connect Mode*, Godot may also change the tiles in the neighboring cells to find the best match. In this example, it changed three of the neighbor tiles.

Note

When you paint on the TileMap in *Connect Mode*, Godot may change tiles in the neighboring cells so that all the peering bits match. But it will **not** change the *terrain* of the neighboring cells. This means that it will only place a tile with the same center bit as the tile that was there before.

See also

For more information about Connect Mode and painting terrains on a TileMap, see Using TileMaps.

Choosing a terrain mode

Godot has three terrain modes: Match Sides, Match Corners, and Match Corners and Sides. The terrain mode determines which peering bits a terrain tile has. This changes what kind of shapes you can make with the tiles, and also how many tiles are needed to paint all the possible shapes.

Note

Terrain modes are similar to Godot 3.x's autotile bitmask modes. Match Corners corresponds to **2x2**, and Match Corners and Sides corresponds to **3x3 minimal**.

While Match Sides does not correspond to a separate autotile mode, a similar sides-matching autotile was possible in Godot 3.x by using a 16-tile subset of 3x3 minimal.

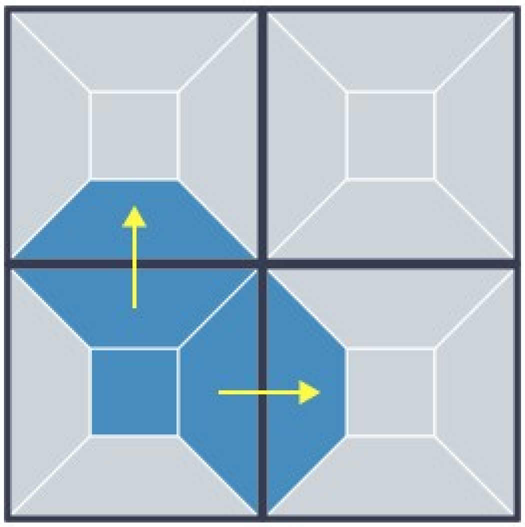
3x3 mode in Godot 3.x, which allowed *individual* diagonal corners to match together and required 256 tiles for a full set, does not have an equivalent in Godot 4.

See also

These tile-matching modes are not unique to Godot. To learn about the theory behind matching tiles in this way, see cr31's Wang Tiles.

Match Sides

- **Peering bits:** Match Sides mode uses peering bits on the tiles' edges. This means that square and isometric tiles have four peering bits, and hexagonal tiles have six. For all shapes, there is only one neighboring tile at each peering bit. Neighbors at the corners do not have any effect.



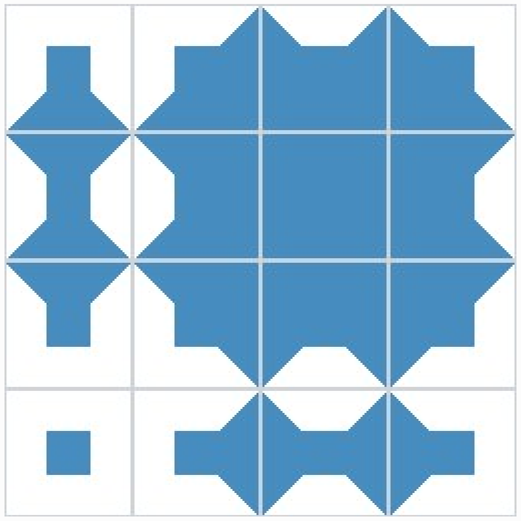
How peering bits match with their neighbors in Match Sides mode.

- **Tiles needed:** 16.
- **Advantages:** These tiles are the easiest to set up and use. You can paint straight lines. You can also paint *either* turns and intersections *or* filled-in rectangles. The top-down example has both uses in the same terrain set.
- **Limitations:** You can't paint diagonal lines. In addition, when you are painting rectangles, there is no difference between outside and inside corners. For many tiles, the artwork requires that inside corners have a different appearance from outside corners. If this is the case, you can't paint shapes that are more complex than single rectangles.

See also

See the alternative tiles example for flexible tiles that do *not* require separate art for inside and outside corners, and can be painted in complex shapes in Match Sides mode.

- **Template:**



*Right-click and choose **Save as...** to download.*

Note

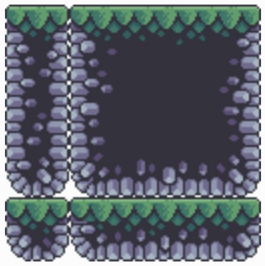
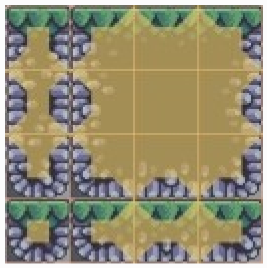
This section includes one or more templates for each terrain mode. You can use them as references for assigning peering bits, or as guides in a paint program. All templates have a square tile size of 64px x 64px, but they are designed to be scaled to other sizes.

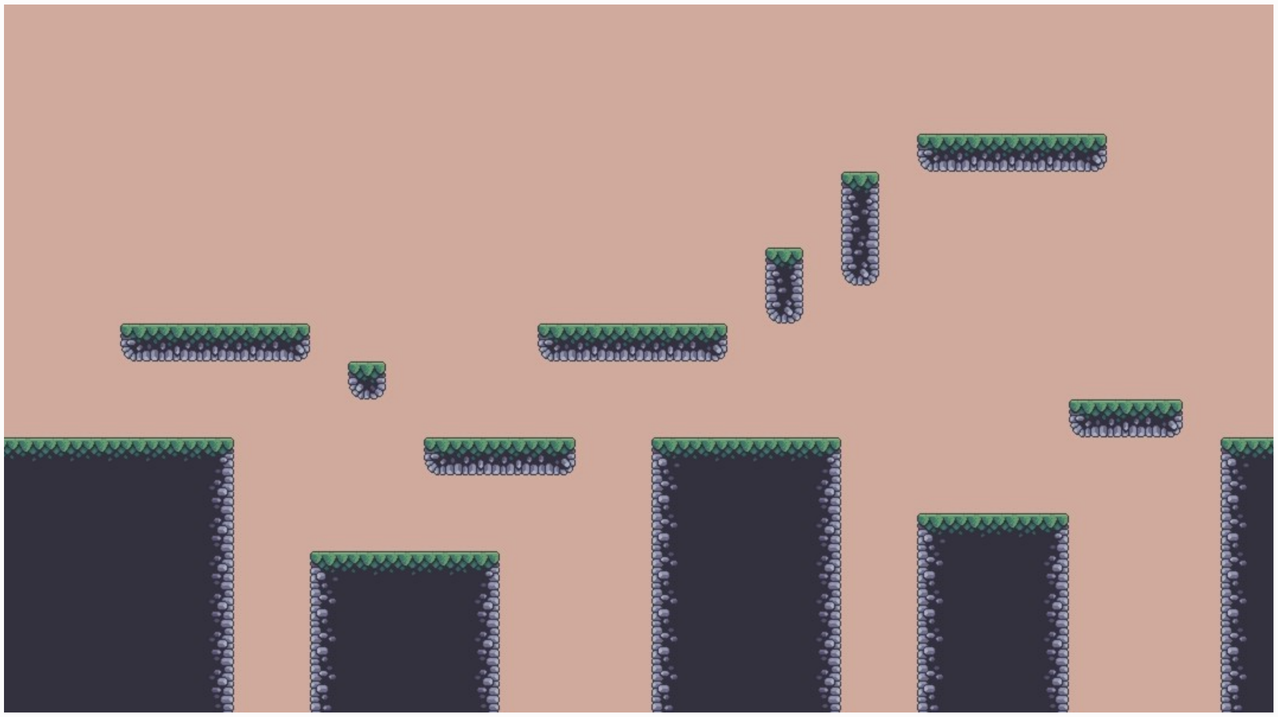
You can download the templates' image files individually from this page. Or you can download the starter project (2d_using_tilesets_terrain_starter.zip) to get all the templates in `png` format.

- **Sidescroller example:** Note the simple rectangles and the straight lines for platforms. For comparison, see the Match Corners and Match Corners and Sides sidescroller examples.

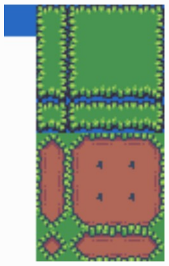
See also

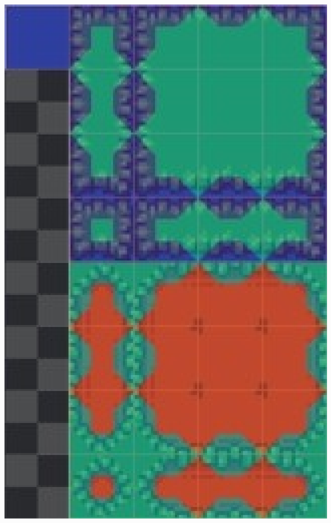
To learn how to create a terrain set and use the bitmasks for tiles like these, see Setting up a new terrain set and Setting up terrain tiles in Paint mode.

Tilesheet
 *Adapted from Treasure Hunters by Pixel Frog (CC0). Right-click and choose **Save as...** to download.*
Bitmask

Level

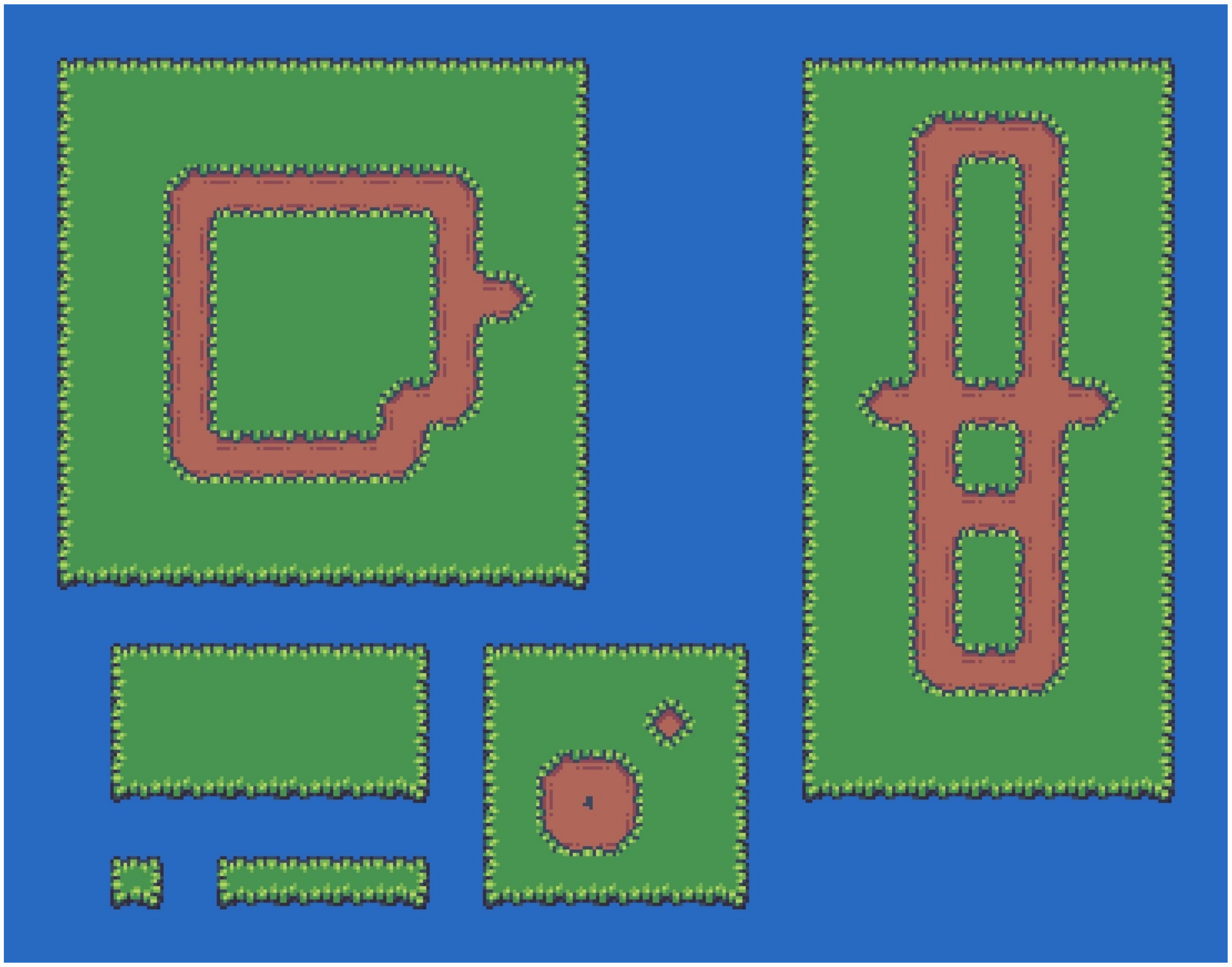


- **Top-down example:** Note the different uses of Match Sides mode for grass and dirt terrains. The grass tiles are painted as rectangular patches of terrain, and the dirt tiles are painted as roads with turns and intersections. For comparison, see the Match Corners and Match Corners and Sides top-down examples.

Tilesheet
 *Adapted from The Field of Floating Islands by Buch (CC0). Right-click and choose **Save as...** to download.*
Bitmask

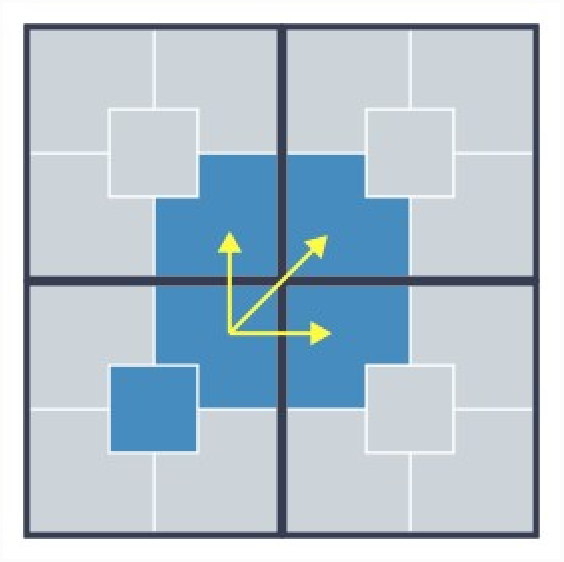


Level



Match Corners

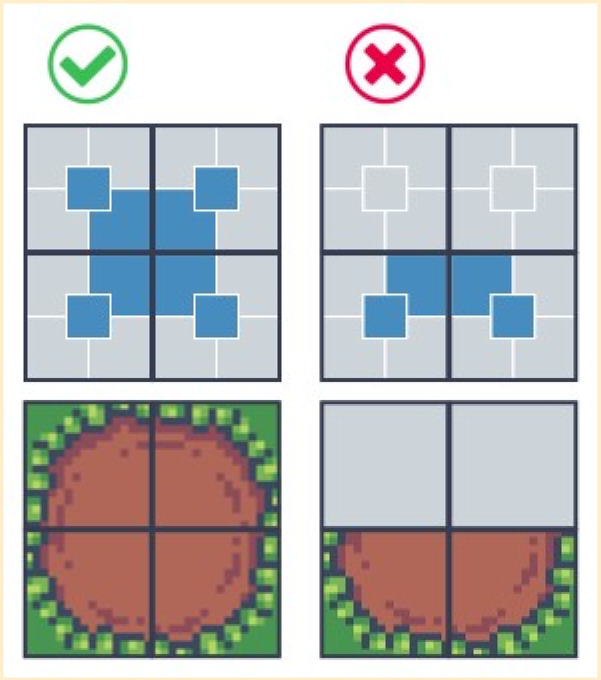
- **Peering bits:** Match Corners mode uses peering bits on the tiles' corners. Square and isometric tiles have four peering bits, and three neighbors at each peering bit. Hexagonal tiles have six peering bits, but only two neighbors at each peering bit. All tiles that share the same corner must have their peering bit set to the same terrain. This means that *square and isometric tiles must be connected in groups of four*, and *hexagonal tiles must be connected in* *groups of three.*


How peering bits match with their neighbors in Match Corners mode.

- **Tiles needed:** 16.
- **Advantages:** You can paint complex shapes. Match Corners is good for painting large patches of landscape or caves.
- **Limitations:** You can only connect tiles together in groups of four (two-by-two blocks). This means that small details and lines only one-tile wide are not possible.

Warning

In Match Corners mode, the corner bits of *all* neighboring tiles must match.

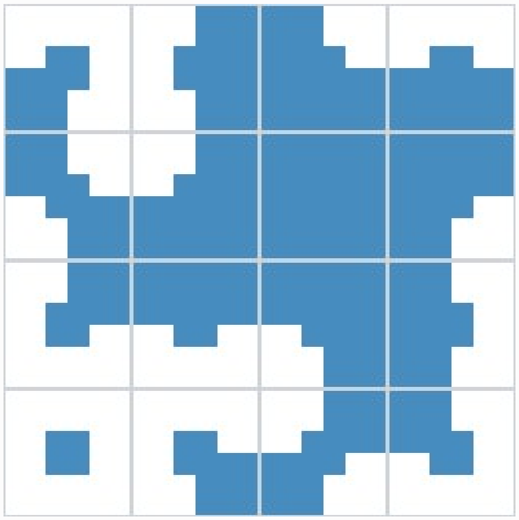


Tip

To prevent errors when painting on the TileMap, try using the rectangle tool. With the rectangle, you can make sure you are always painting at least four tiles of the same terrain together.

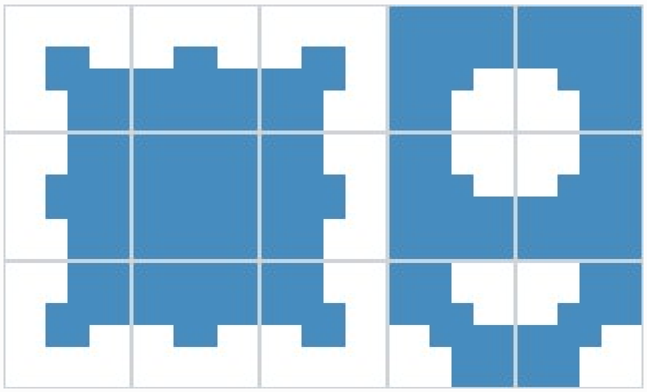
In procedural generation, you will need to make sure that your script doesn't assign fewer than four cells together to a terrain. If you're using pixel values from noise to set terrains, you can scale up the noise x2 to make sure that all terrains are painted in groups of four.

- **Templates:** In this first template for Match Corners mode, all the tiles connect to each other.



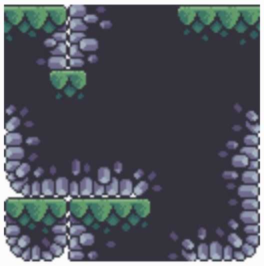
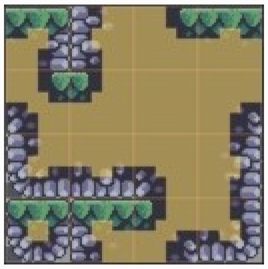
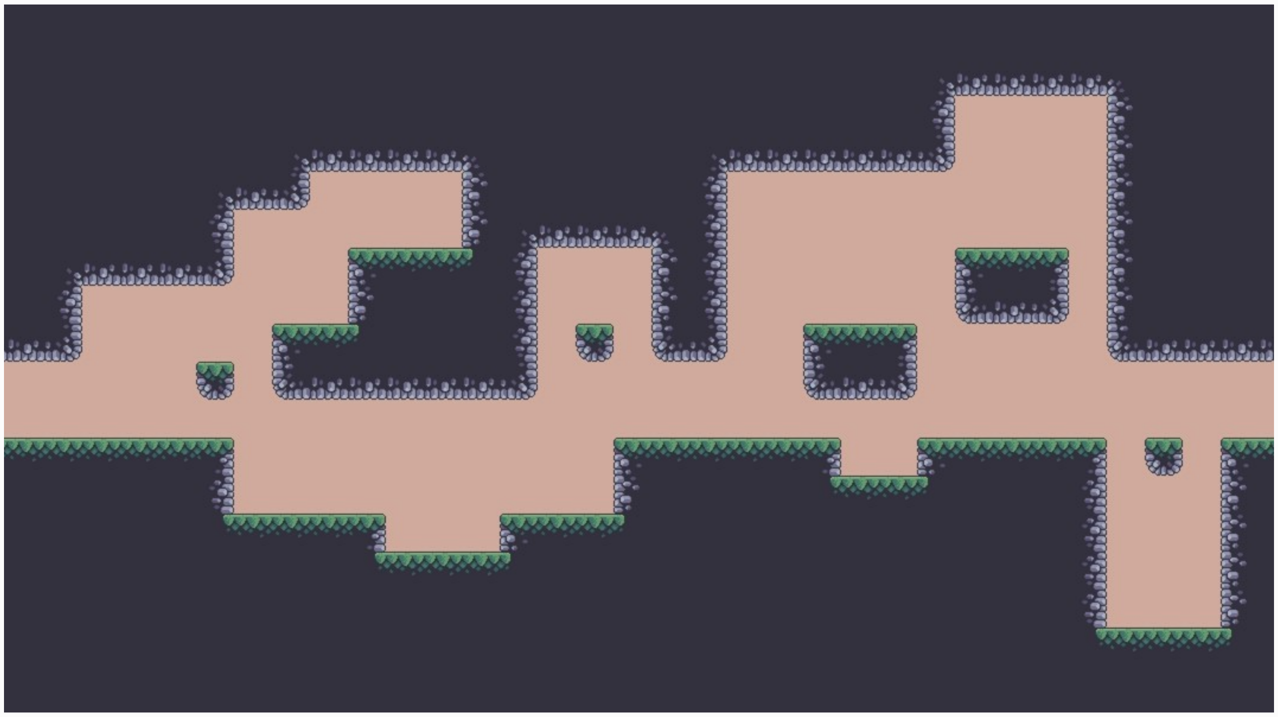
*Right-click and choose **Save as...** to download.*

In this alternate template, the tiles are arranged according to outside and inside corners. Publicly available tilesheets are frequently set up like this. Note that this template only has 15 tiles: it is missing the single tile.



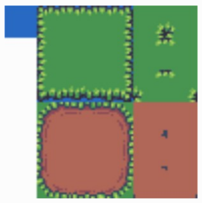
*Right-click and choose **Save as...** to download.*

- **Sidescroller example:** Note the complex shapes of the caves in this level. There are also small single-tile platforms, but no long, thin platforms. For comparison, see the Match Sides and Match Corners and Sides sidescroller examples.

Tilesheet
 *Adapted from Treasure Hunters by Pixel Frog (CC0). Right-click and choose **Save as...** to download.*
Bitmask

Level


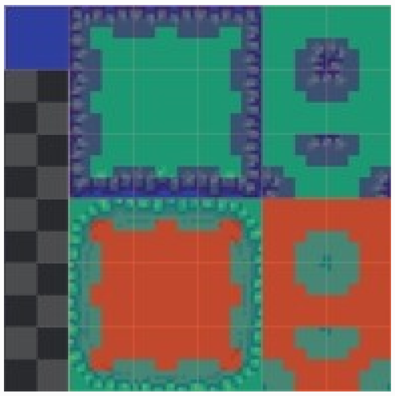
- **Top-down example:** Note the large patches of terrain in this level, but the lack of single-tile roads or small details. For comparison, see the Match Sides and Match Corners and Sides top-down examples.

Tilesheet

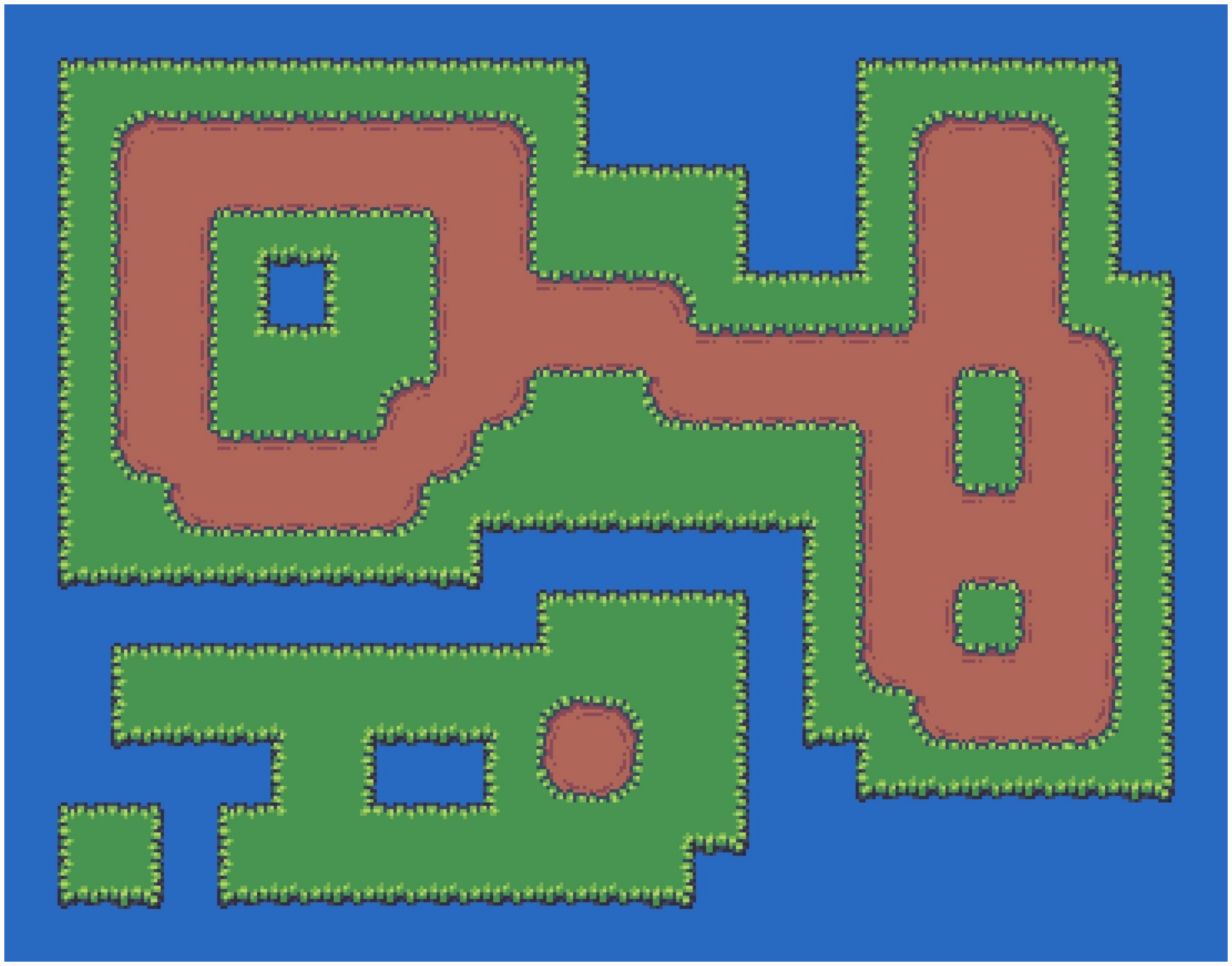


*Adapted from The Field of Floating Islands by Buch (CC0). Right-click and choose **Save as...** to download.*

Bitmask

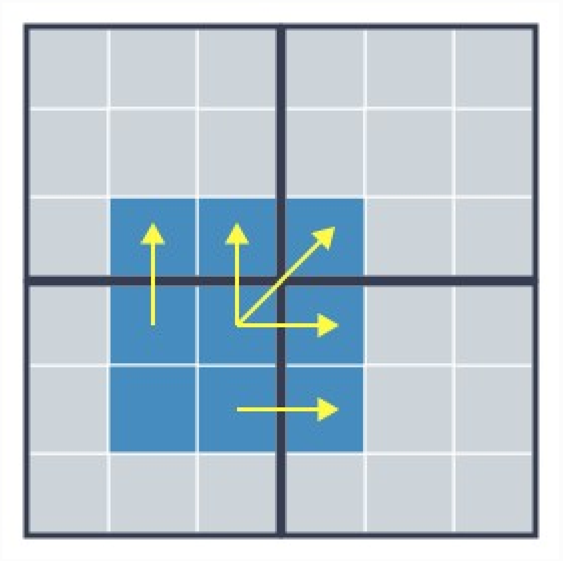


Level



Match Corners and Sides

- **Peering bits:** Match Corners and Sides mode uses peering bits on the tiles' corners and edges. Square and isometric tiles have 8 peering bits, and hexagonal tiles have 12. As in the other modes, tiles have only one neighbor at each side, while square and isometric tiles have three neighbors at each corner, and hexagonal tiles have two. All tiles that share the same corner must have that corner's peering bit set to the same terrain.



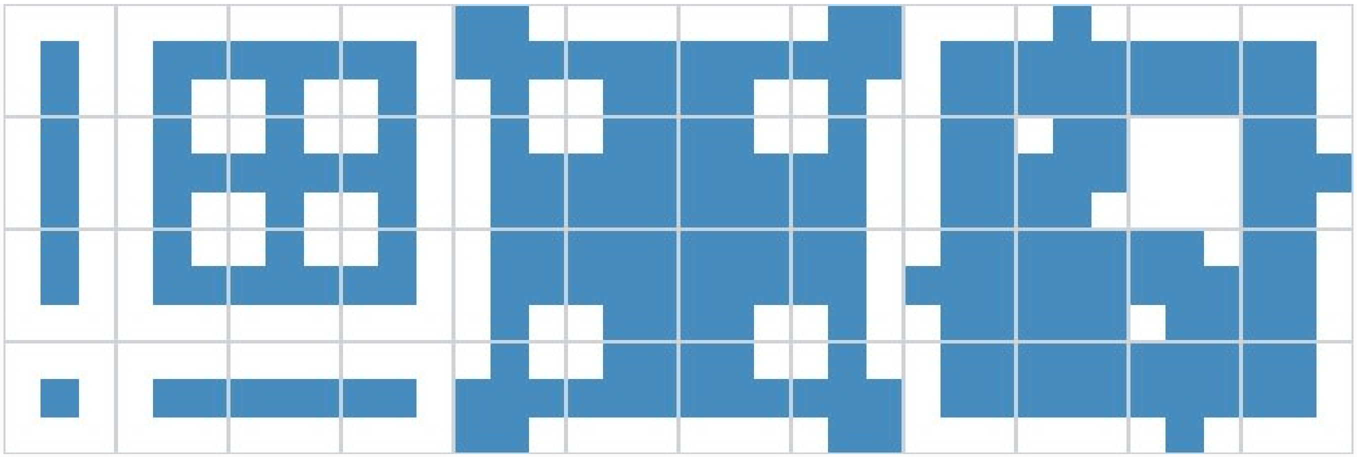
How peering bits match with their neighbors in Match Corners and Sides mode.

- **Tiles needed:** 47.
- **Advantages:** This is the most versatile terrain mode. You can draw all the shapes from the other modes, but without their limitations. You can create single-tile paths, complex regions of terrain, and any needed twists or turns between them.
- **Limitations:** It requires more tiles than the other modes. But there is only one limitation for shapes: you cannot create diagonal lines.

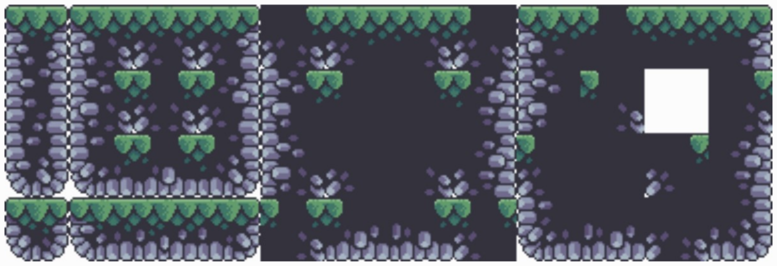
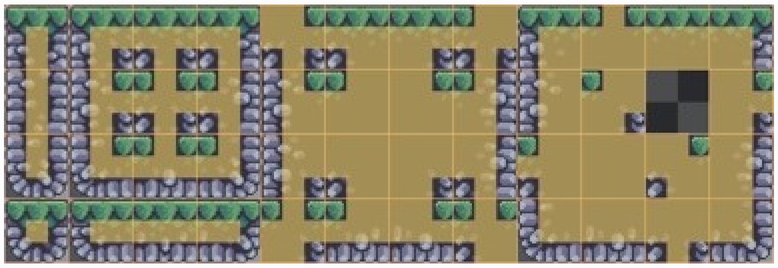
See also

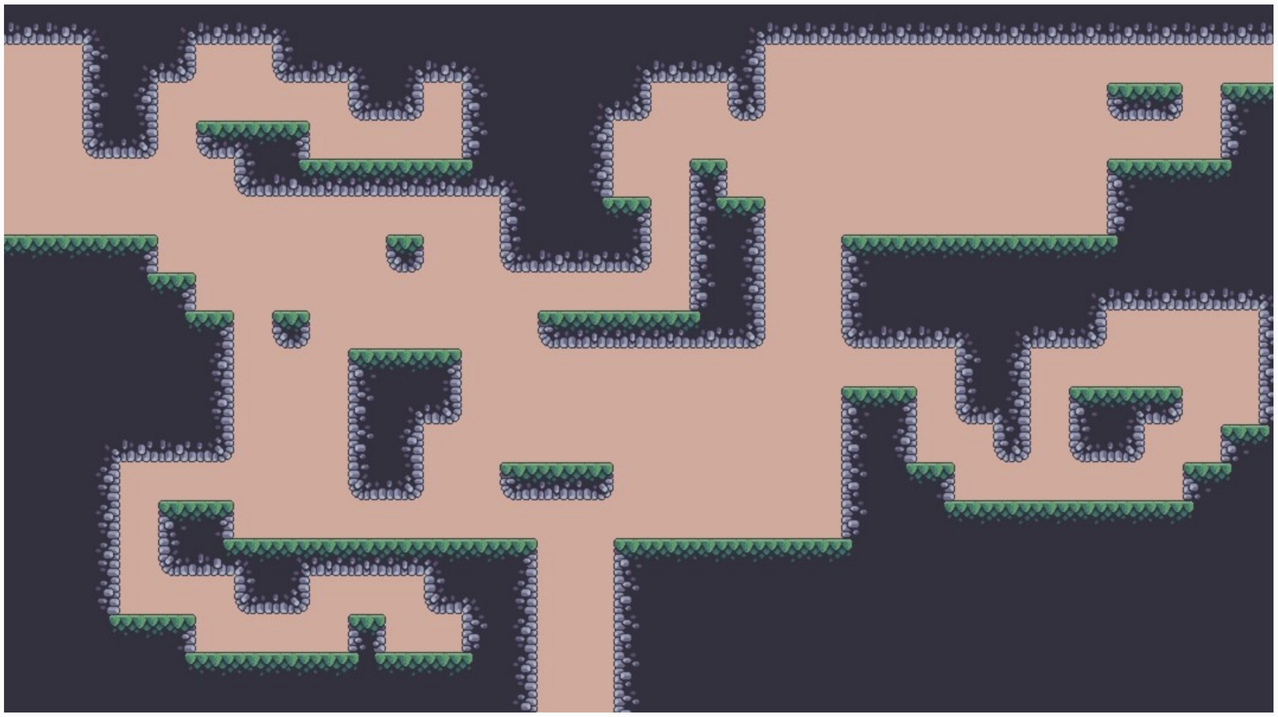
Free tools like Webtyler and TilePipe2 can automatically generate the full 47 tiles for Match Corners and Sides mode from only a few tiles.

- **Templates:**


*Right-click and choose **Save as...** to download.*

- **Sidescroller example:** Note that complex caves and narrow platforms are possible in the same TileMap. For comparison, see the Match Sides and Match Corners sidescroller examples.

Tilesheet
 *Adapted from Treasure Hunters by Pixel Frog (CC0). Right-click and choose **Save as...** to download.*
Bitmask

Level

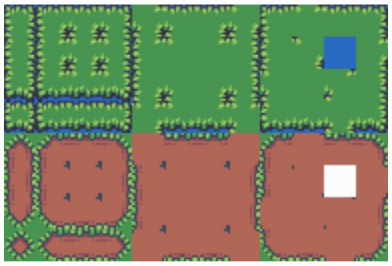


- **Top-down example:** Note the complex terrain shapes, as well as roads and small details. For comparison, see the Match Corners and Match Corners and Sides top-down examples.

Note

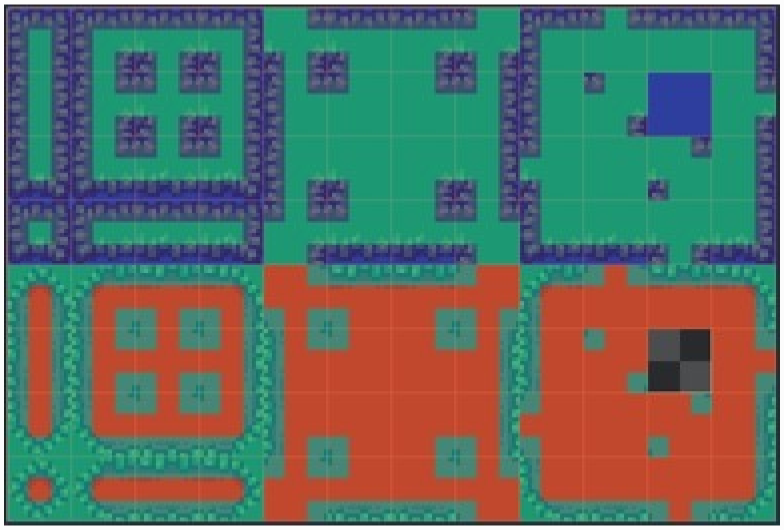
This tilesheet was automatically generated from the original source using Webtyler. To try it yourself, you can use the *minitile* in the starter project (2d_using_tilesets_terrain_starter.zip).

Tilesheet

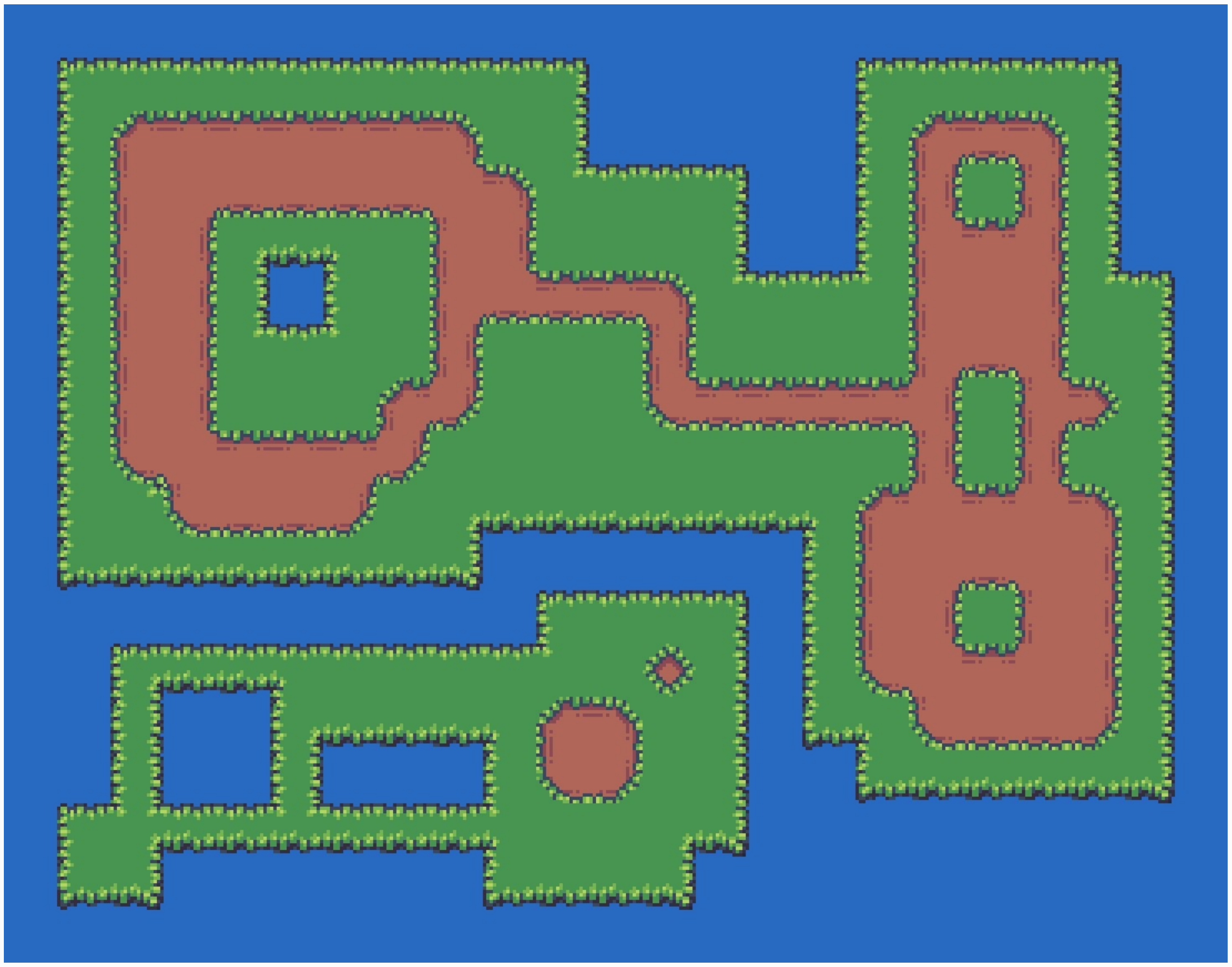


*Adapted from The Field of Floating Islands by Buch (CC0). Right-click and choose **Save as...** to download.*

Bitmask



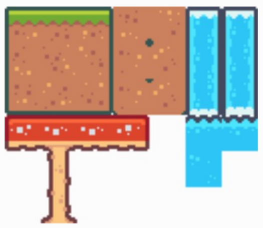
Level



Setting up a new terrain set

In this section, we will create a terrain set for the ground and mushroom tiles in the Pixel Platformer tilesheet. (We will set up the water tiles later in Animating terrain tiles.)

Pixel Platformer tilesheet


*Adapted from Pixel Platformer by Kenney (CC0). Right-click and choose **Save as...** to download.*

The ground tiles match by their corners, so they will need Match Corners mode, and the mushroom tiles match according to their sides, so they will need Match Sides mode. A terrain set can only have one mode, so we will create two separate terrain sets.

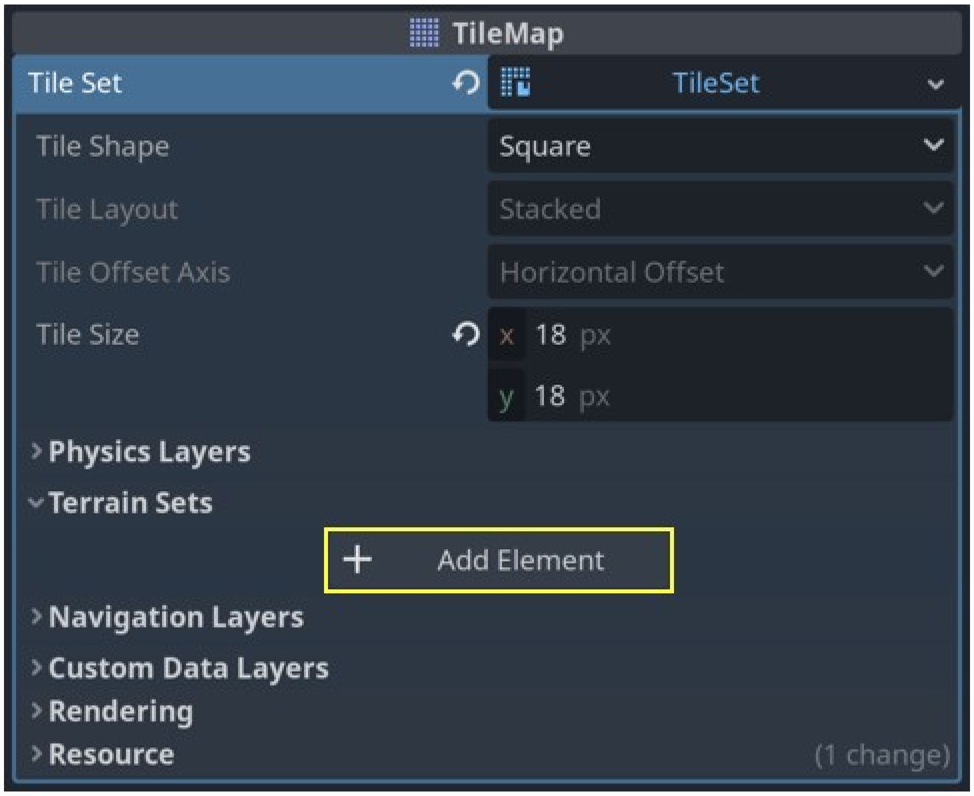
Caution

Tiles in different terrain sets cannot match with each other, since they use different peering bits. In this case, it doesn't matter, since we will be painting the ground and mushrooms separately over a background of empty cells.

If we needed to paint them both over the same background terrain, or if they needed to transition to each other, we would have to put them in the same terrain set. We could do this using Match Corners and Sides mode. We could automatically generate the extra tiles we need using the tools from the Match Corners and Sides section, or we could reuse tiles using alternative tiles.

To create a terrain set, select the **TileMap** node, go to the inspector, and open the **TileSet** *resource*. (See Creating a new TileSet.)

Open the **Terrain Sets** section and press the **Add Element** button.

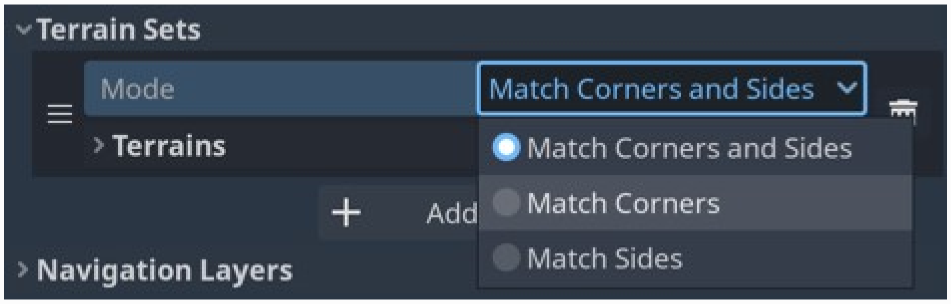

TileMap

Tile Set	TileSet
Tile Shape	Square
Tile Layout	Stacked
Tile Offset Axis	Horizontal Offset
Tile Size	x 18 px, y 18 px

Physics Layers
Terrain Sets
Add Element
Navigation Layers
Custom Data Layers
Rendering
Resource (1 change)

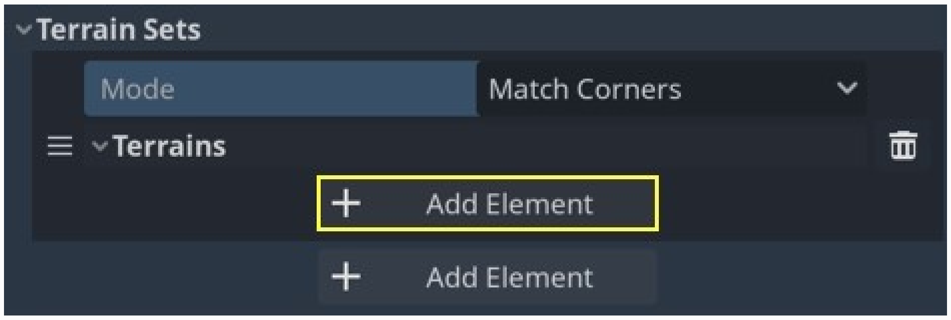
Creating a terrain set in the TileSet resource inspector (within the TileMap node).

Use the **Mode** dropdown menu to choose a terrain mode. We will use Match Corners mode for this terrain set.



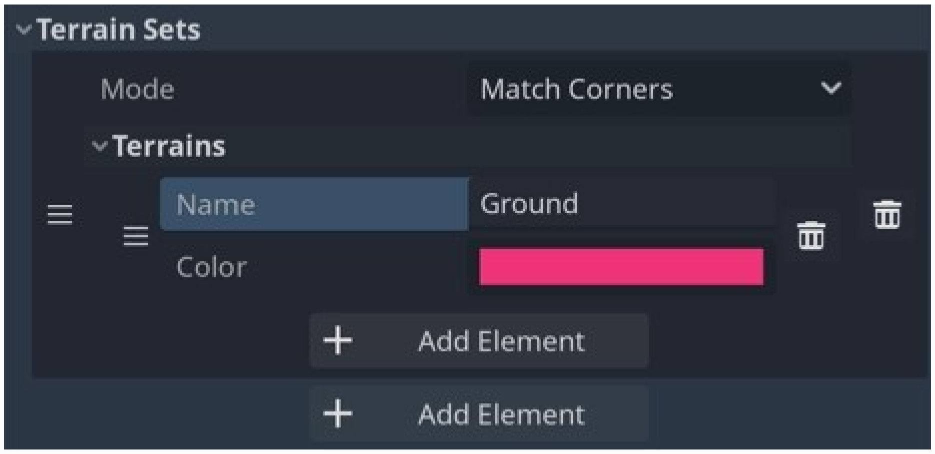
Selecting a terrain mode.

Open the **Terrains** section and press the **Add Element** button to add a new terrain.



Adding a terrain.

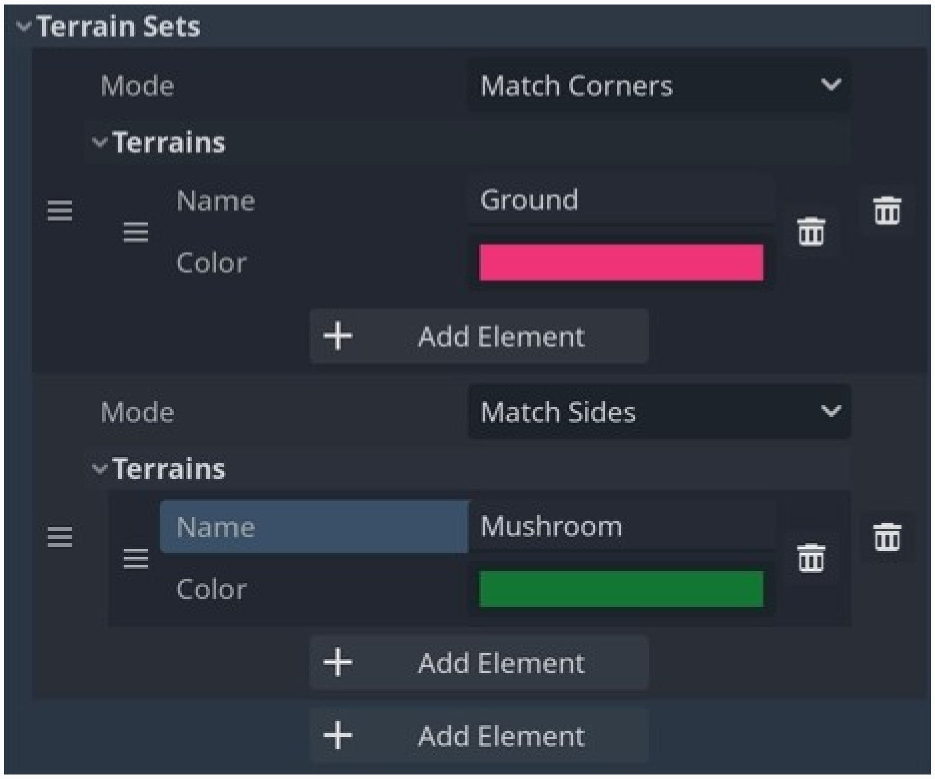
You can then edit the terrain's name and color. We will use this terrain for our ground tiles, so we'll call it `Ground`.



Editing a terrain's name and color.

> **Tip**
>
> A terrain's color will be used as an overlay for the tiles' bitmasks. If the bits are hard to see, you can change this color at any time. A bright color that contrasts with the tile's texture is often easiest to see.

Follow the same steps to create a second terrain set. This time, choose Match Sides mode and add the `Mushroom` terrain.

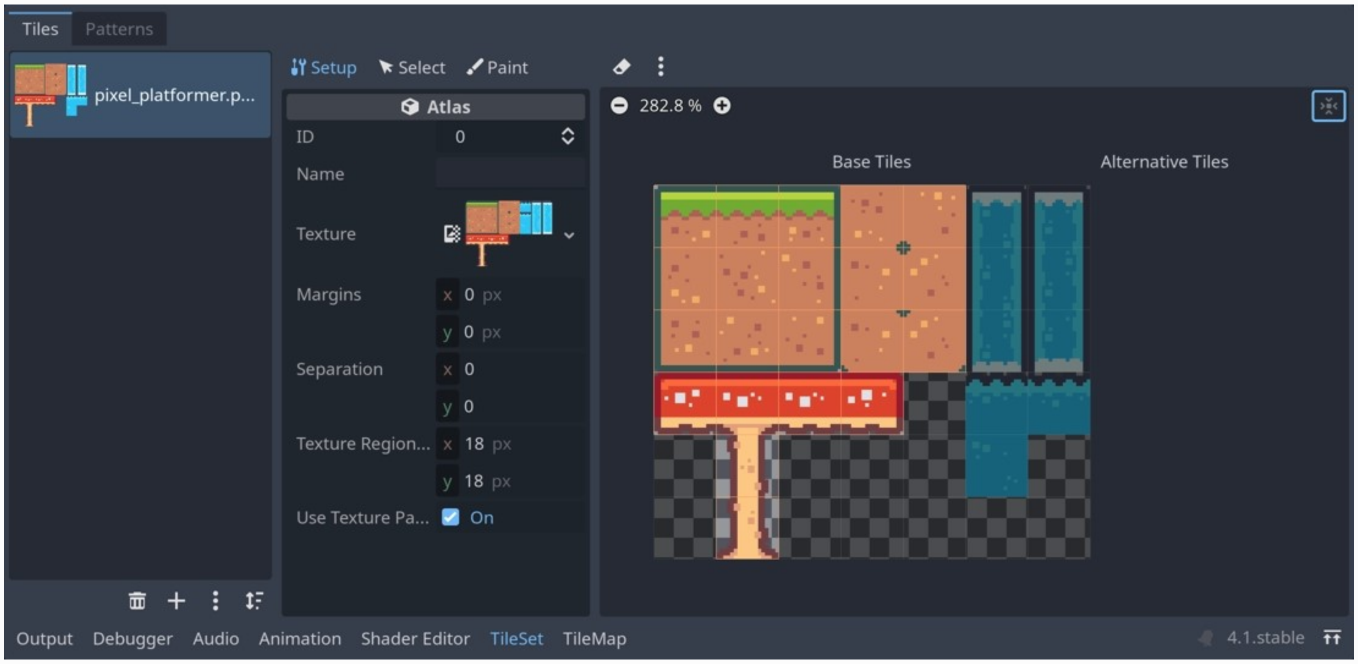


Setting up the second terrain set.

Next, we need to set up the tiles with terrain properties. We will learn how to do this in Paint mode and in Select mode here. You can also access these properties from a script using the TileData class.

Setting up terrain tiles in Paint mode

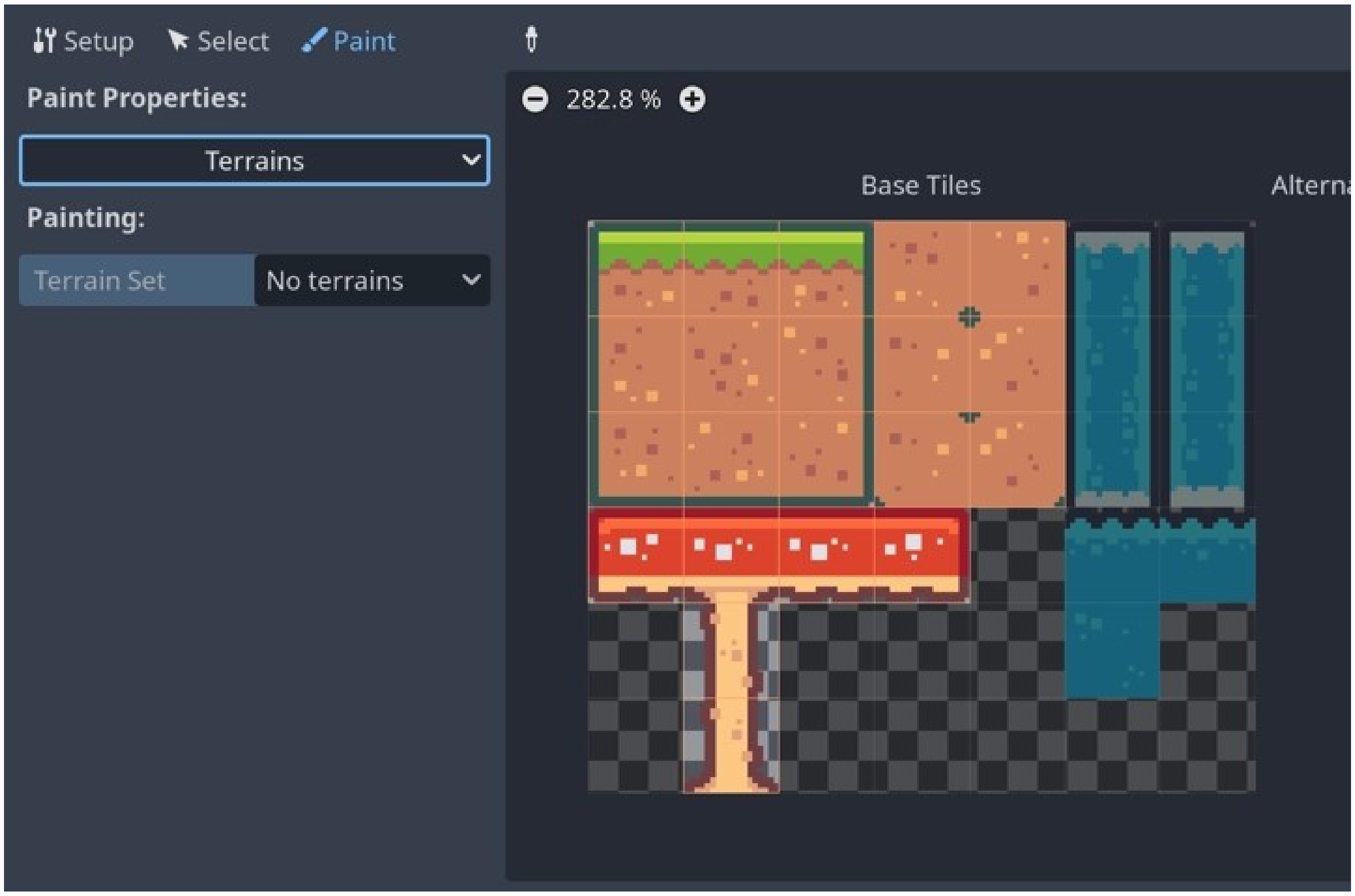
First, open the **TileSet editor** in the bottom panel, open **Setup** mode, and add the ground and mushroom tiles. (See Creating a new TileSet: Using a tilesheet).



ID: 0
Name:
Texture:
Margins: x 0 px, y 0 px
Separation: x 0, y 0
Texture Region...: x 18 px, y 18 px
Use Texture Pa...: On

Creating tiles in Setup mode.

Open **Paint** mode by clicking the button at the top. Then open the **Paint Properties** dropdown menu and choose **Terrains**.



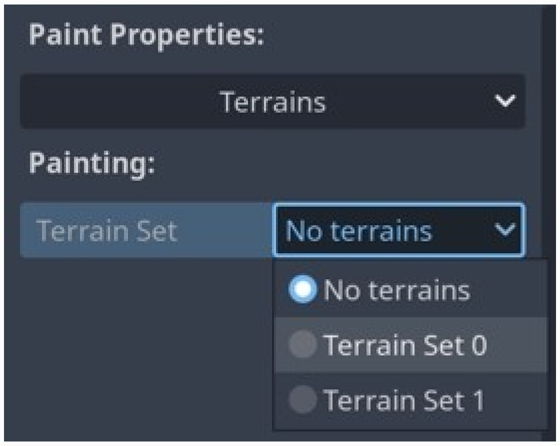
Paint Properties: Terrains
Painting:
Terrain Set: No terrains

Choosing Terrains under Paint Properties.

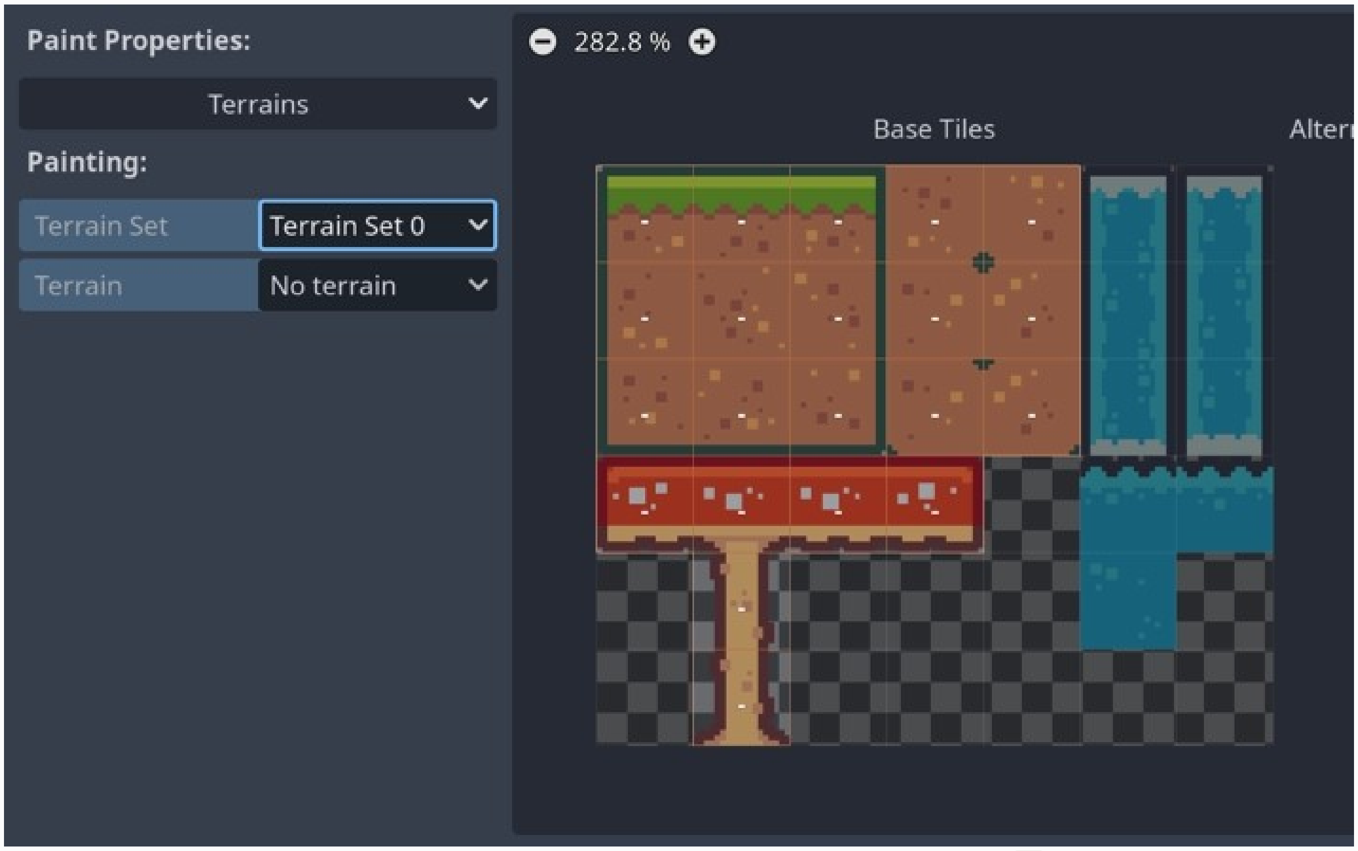
Before you can assign terrains, you must first assign a terrain set. Choose `Terrain Set 0` for the ground tiles.

Note

Terrain sets are listed according to their order in the TileSet, starting with an index of `0`.



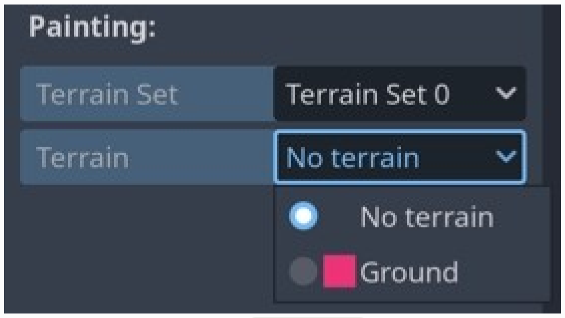
Choosing a terrain set.



Tiles with no terrain sets assigned have an overlay of `-`.

Click and drag to paint the ground tiles with the terrain set.

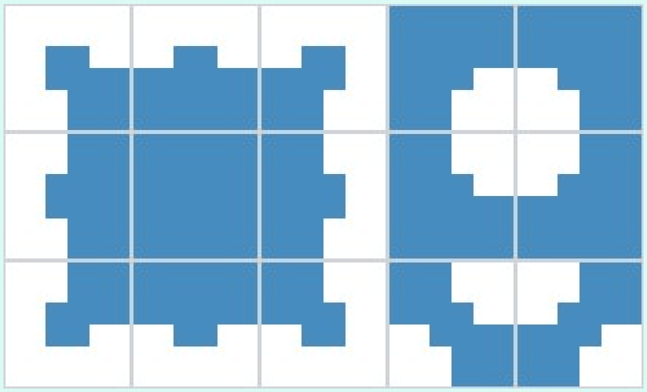
Then open the **Terrain** dropdown and select the `Ground` terrain.



Selecting the `Ground` terrain.

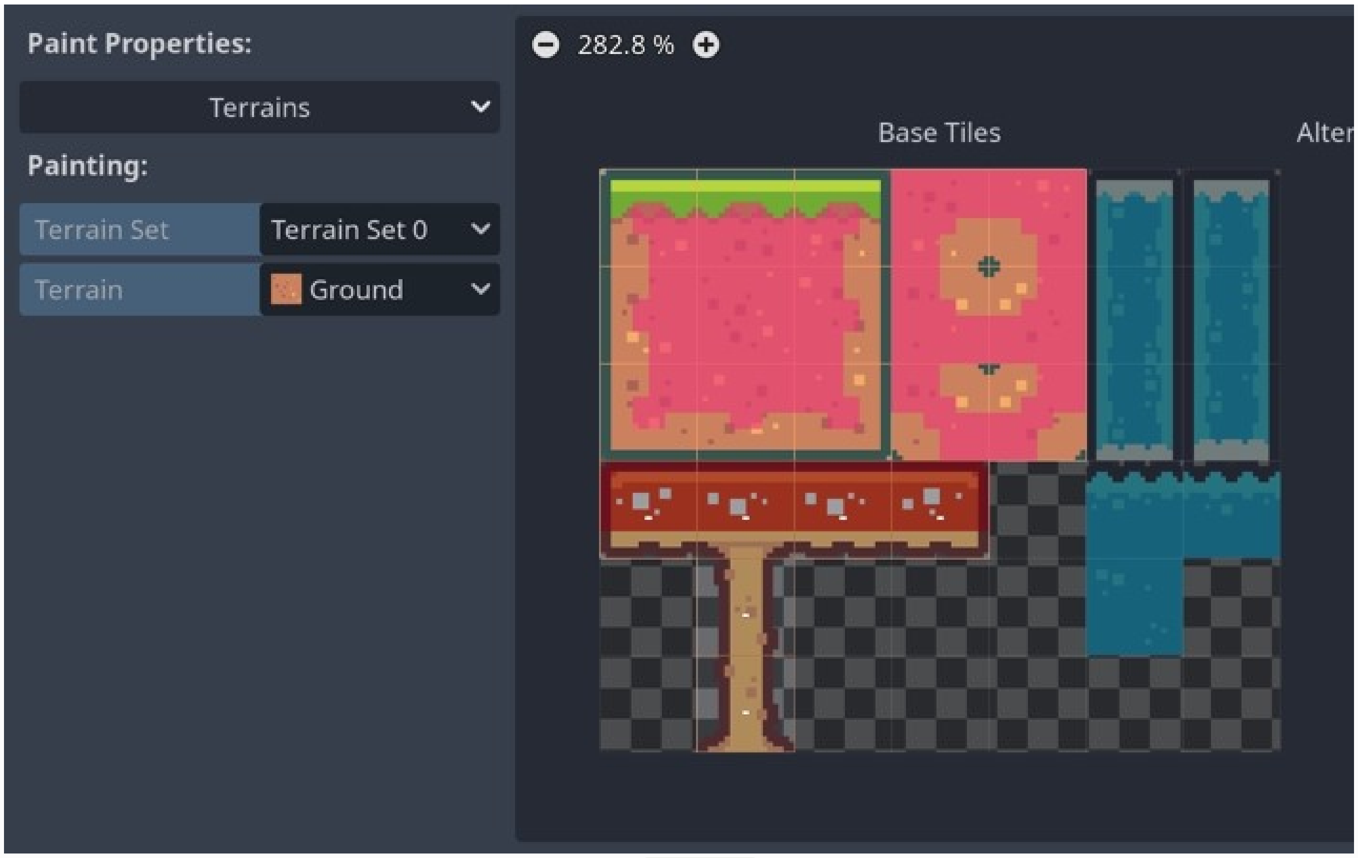
Tip

Sometimes it is hard to tell which peering bits to paint based on the tiles' texture alone. Using a reference, like the templates, can be helpful. In this example, the ground tiles are arranged according to the following template:



Alternate template for Match Corners mode.

Using the template as a guide, paint the tiles with the `Ground` terrain.



Painting the `Ground` terrain.

Later, when we paint the `Ground` terrain on the TileMap, we will be painting it on an empty background. We will leave the peering bits that don't match to other ground tiles empty so they will match to empty space.

Warning

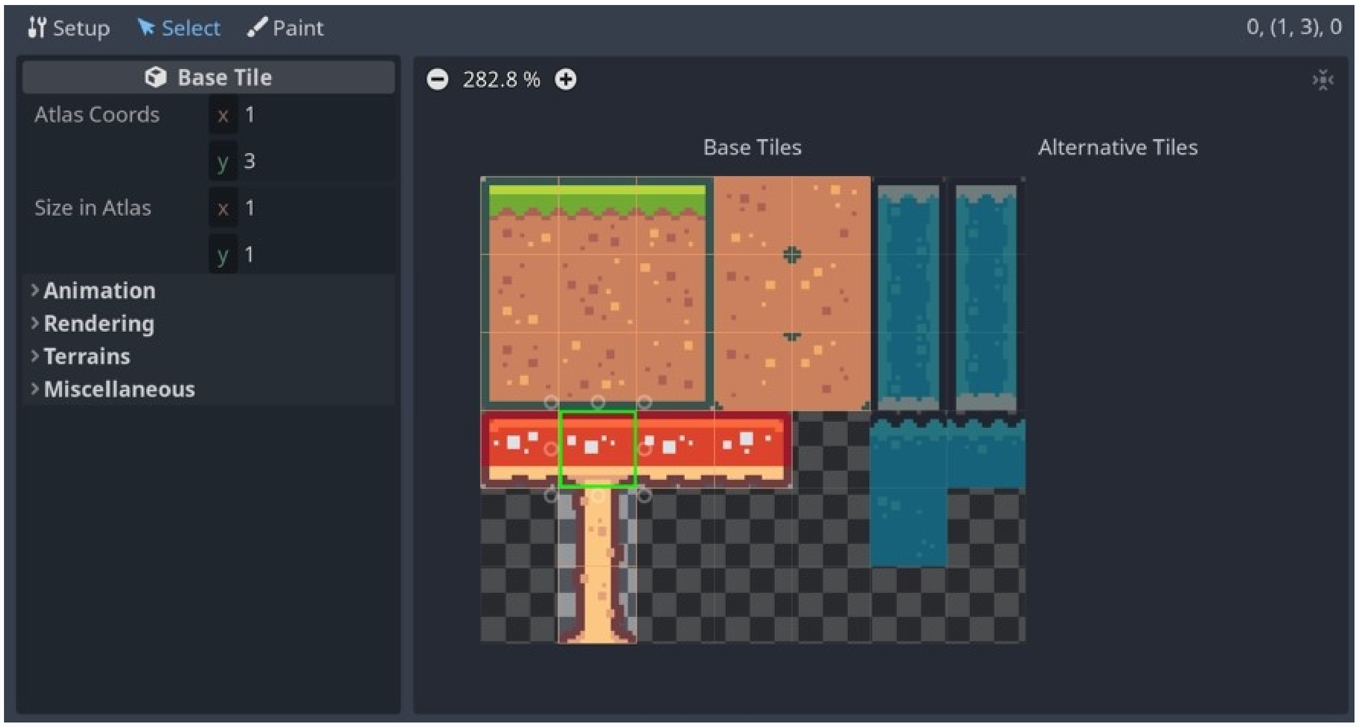
Note that we've assigned the `Ground` terrain to all the ground tiles' center bits. This means that when we paint on TileMap with the `Ground` terrain, Godot will choose one of these tiles to place.

If you leave a tile's center bit empty, Godot will have to guess what terrain the tile belongs to. This can lead to unexpected results, so it is not recommended.

Normally, we would now paint the mushroom tiles with `Terrain Set 1` and the `Mushroom` terrain. For this example, however, we are going to set them up in Select mode.

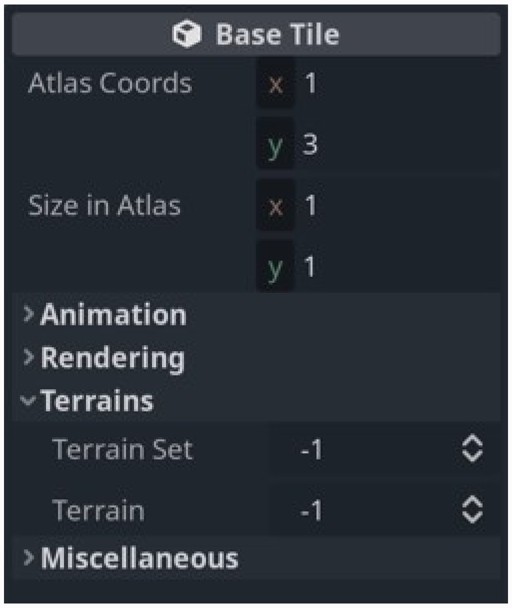
Setting up terrain tiles in Select mode

In the **TileSet editor**, open **Select** mode. Then left-click on one of the mushroom tiles to select it.



Selecting a tile.

Expand the **Terrains** section.



Base Tile

Atlas Coords x 1 y 3

Size in Atlas x 1 y 1

Animation

Rendering

Terrains

Terrain Set -1

Terrain -1

Miscellaneous

Expanding the Terrains section.

The tile we selected does not yet have a terrain set or terrain assigned, so both values are set to a default of `-1`. `-1` means "no terrain set" or "no terrain".

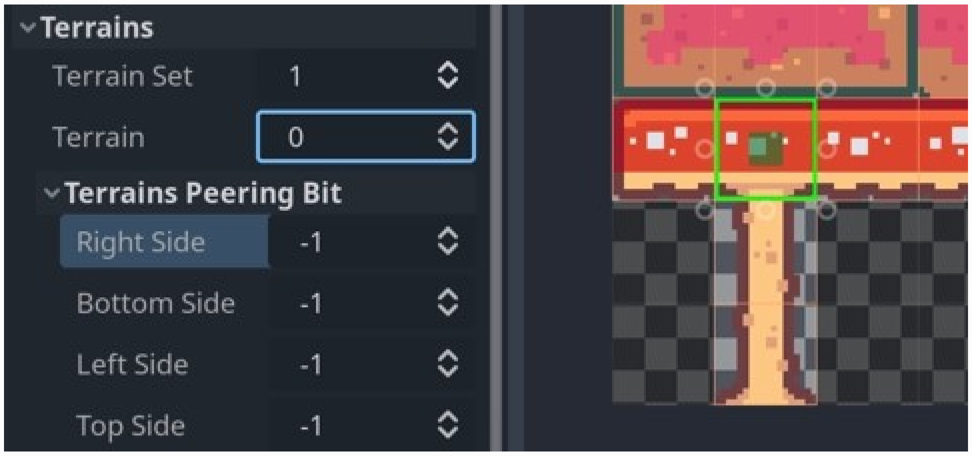
> **Note**
>
> In Paint mode, we could select the terrain set and terrain from dropdown menus. But in Select mode (or in scripting), we need to know the exact values for each one.

A *terrain set*'s index is based on its order inside the *TileSet*. The first terrain set has an index of `0`, not `1`. A *terrain*'s index is based on its order inside its *terrain set*, with numbering also starting from `0`.

In this example, the `Ground` terrain has a terrain set of `0` and a terrain of `0`. The `Mushroom` terrain has a terrain set of `1` and a terrain of `0`. They have the same terrain index since the numbering is independent for each terrain set.

Since this tile belongs to the `Mushroom` terrain, set the **Terrain Set** to `1` and the **Terrain** to `0`. *Terrain* is the same as the center bit that we assigned in Paint mode.

Once you set these values, the **Terrains Peering Bit** section appears.


Terrains
Terrain Set: 1
Terrain: 0
Terrains Peering Bit
Right Side: -1
Bottom Side: -1
Left Side: -1
Top Side: -1

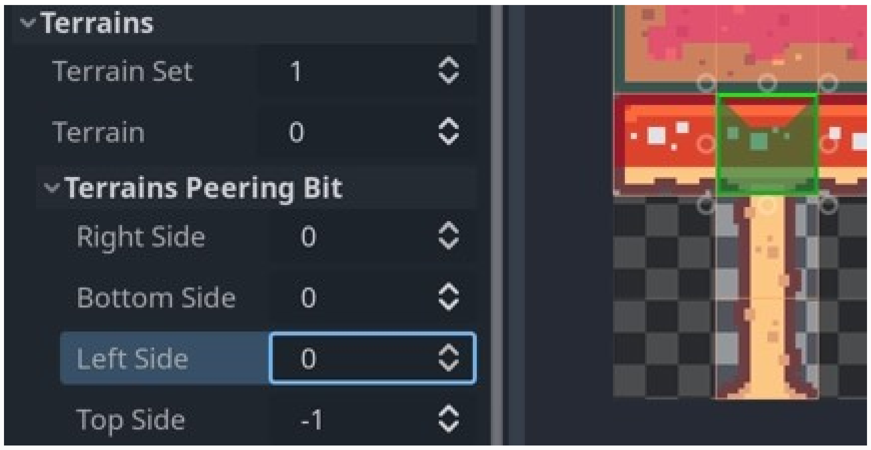
The terrains peering bit section appears.

By default, all the peering bits are set to a *terrain index* of `-1`. In the same way that we painted terrains onto peering bits in Paint mode, we now need to assign terrain indexes to them.

Note

For peering bits, `-1` is a special value which refers to empty space. Setting a peering bit to `-1` is the same as leaving it empty when painting a bitmask.

The `Mushroom` terrain's index is `0`. We should set the peering bits to `0` in directions where we want other mushroom tiles placed. Because we will be painting the mushroom tiles on a background of empty space, we can leave any other bits set to `-1`.


Terrains

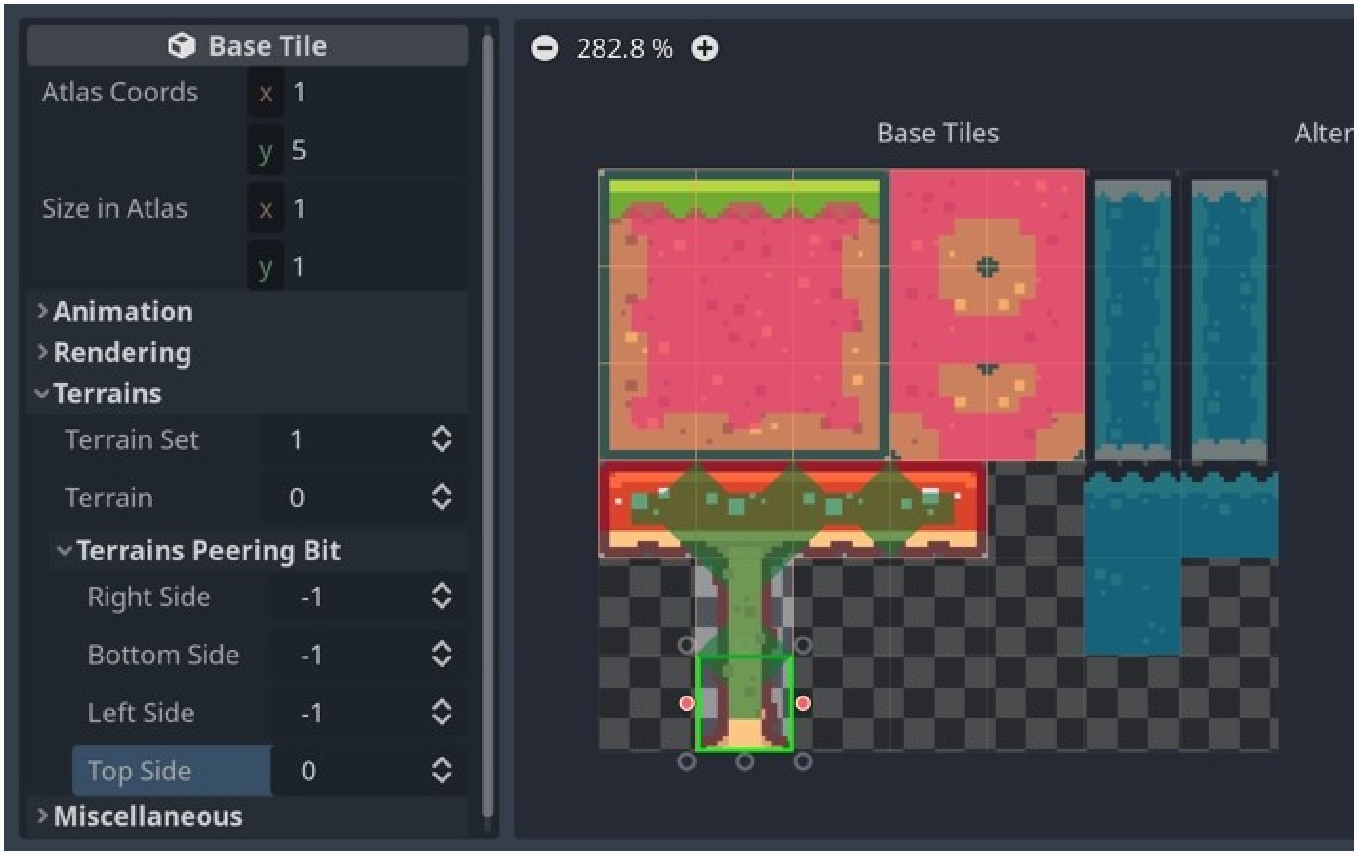
Terrain Set	1
Terrain	0

Terrains Peering Bit

Right Side	0
Bottom Side	0
Left Side	0
Top Side	-1

When peering bits are set, they appear as an overlay in the color of their terrain.

Finish setting the terrain set, terrain, and peering bits for the other mushroom tiles.



Base Tile

Atlas Coords	x	1
	y	5
Size in Atlas	x	1
	y	1

Animation
Rendering
Terrains

Terrain Set	1
Terrain	0

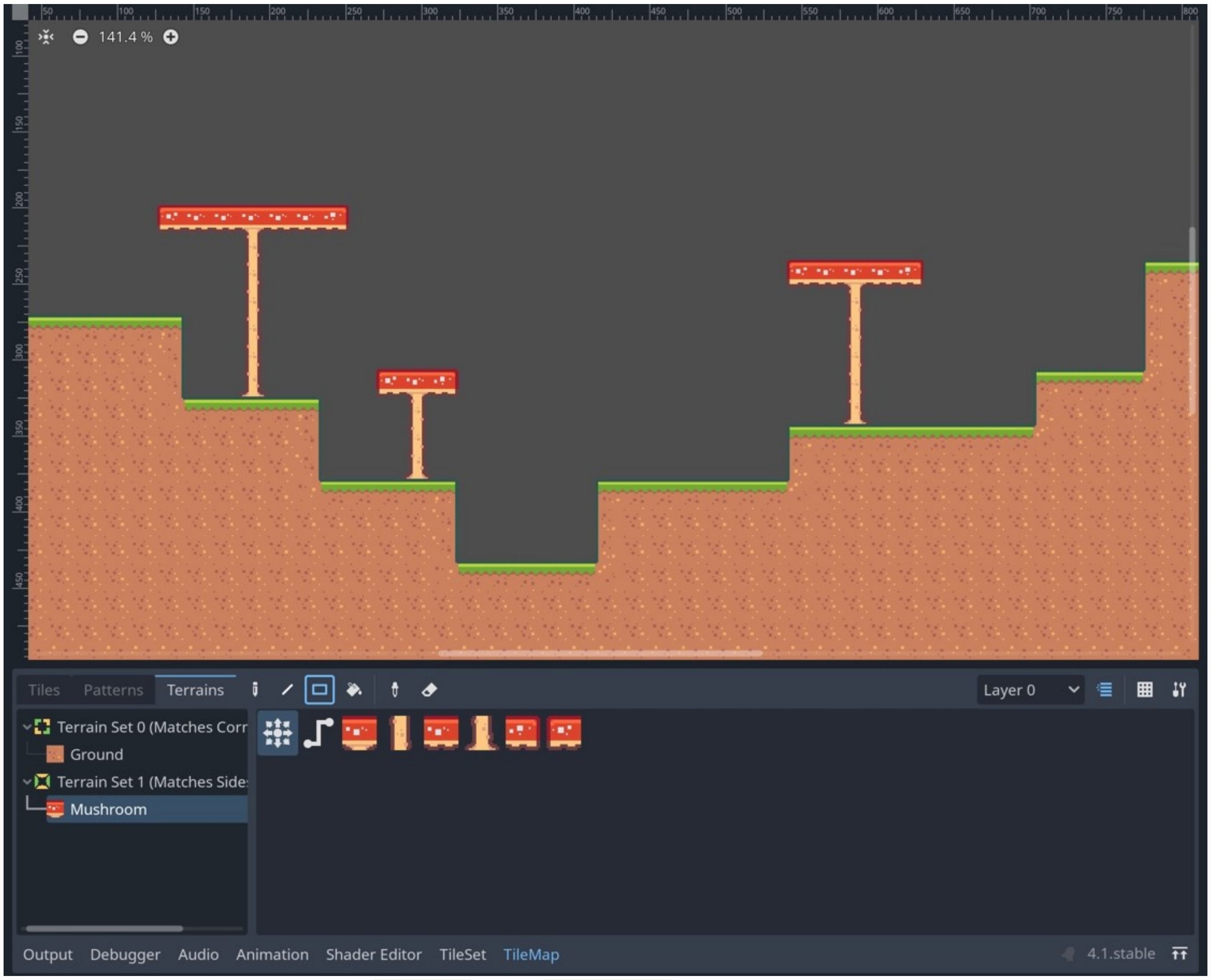
Terrains Peering Bit

Right Side	-1
Bottom Side	-1
Left Side	-1
Top Side	0

Miscellaneous

Finished bitmask.

You can now open the **TileMap editor** and paint the `Ground` and `Mushroom` terrains. (See: Using TileMaps).



Painting the `Ground` and `Mushroom` terrains on the TileMap.

Special cases

See also

The examples in this section assume you are already familiar with the following concepts. If you're not, you can review their sections first:

- Creating a new TileSet
- Setting up a new terrain set
- Setting up terrain tiles in Paint mode

Animating terrain tiles

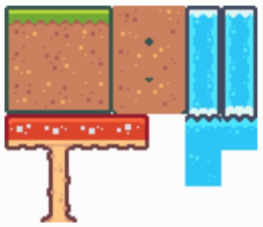
Terrain tiles are atlas tiles that have terrain properties assigned. This means you can also assign other properties of atlas tiles to them, including *animation*.

We will use the TileSet from Setting up a new terrain set, with the ground and mushroom tiles already set up.

Note

To start from this point, download the starter project (2d_using_tilesets_terrain_starter.zip) and open the `setting_up_example` scene.

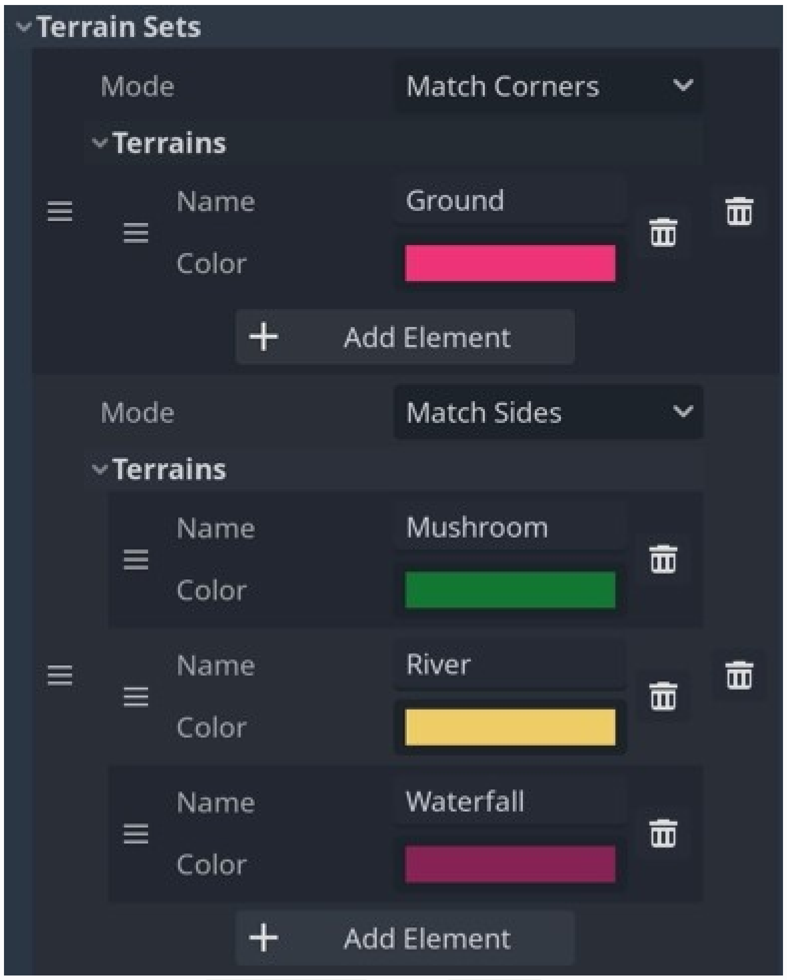
Pixel Platformer tilesheet



Adapted from Pixel Platformer by Kenney (CC0). Right-click and choose ***Save as...*** *to download.*

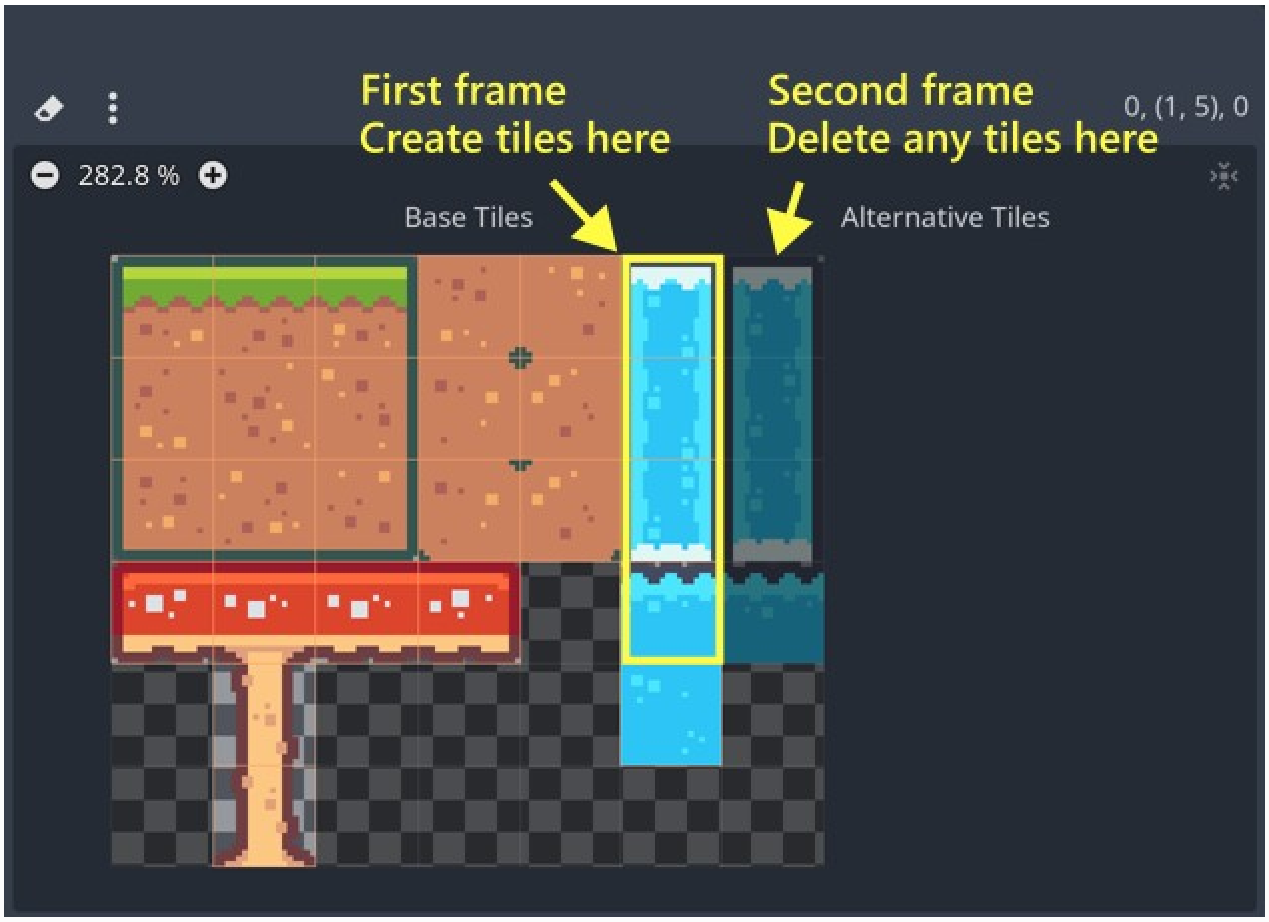
The waterfall and river tiles have a two-frame animation arranged in two columns. The bottom river tile only has one texture, so it will not be animated.

They both need to matched according to their sides, so we can add them to the Match Sides terrain set we already created for the mushroom tiles.



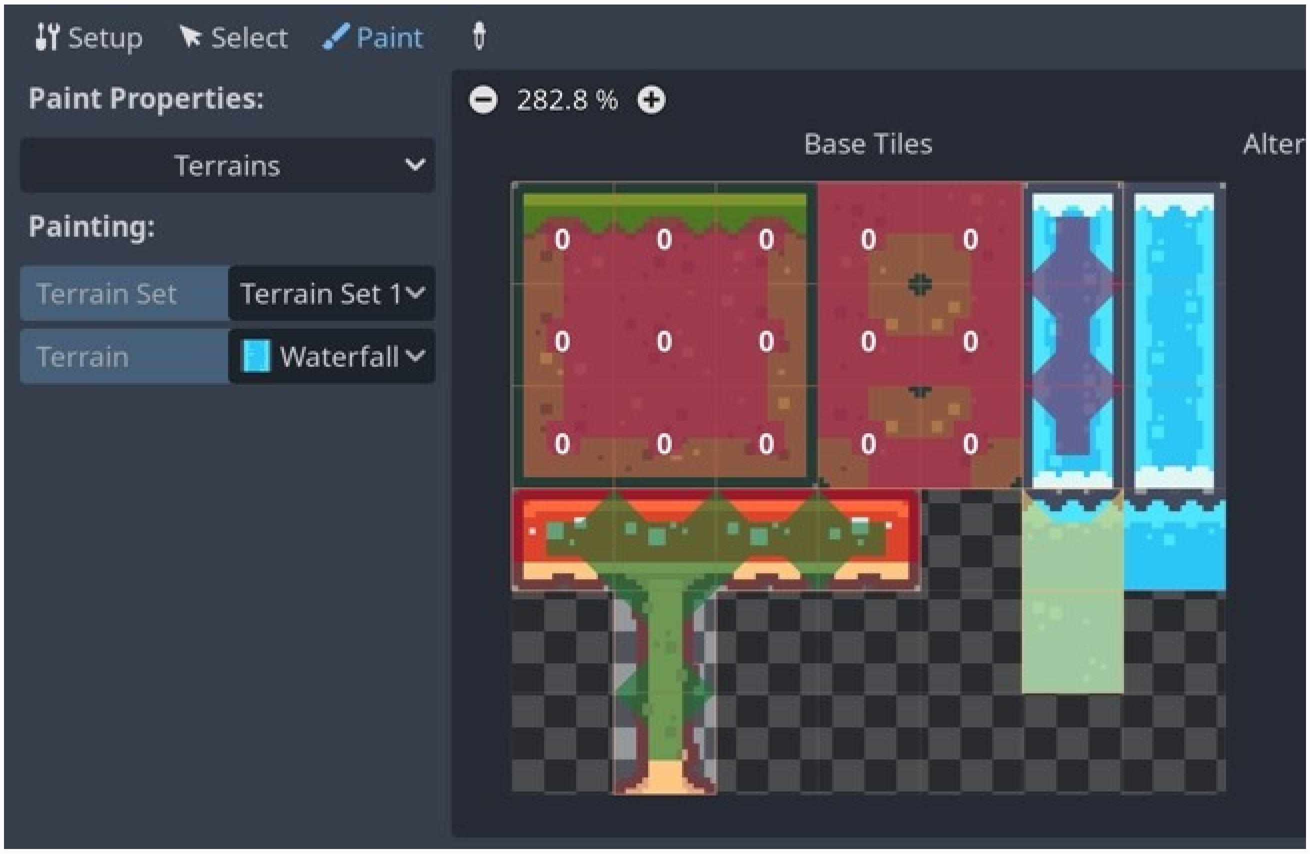
Adding `Waterfall` and `River` terrains.

Go to the **TileSet editor**. In the **Setup** tab, create new tiles in the column for the *first frame* of the animation only. If there are already tiles created in the column for the *second frame*, **right-click** on each of them and press **Delete**.



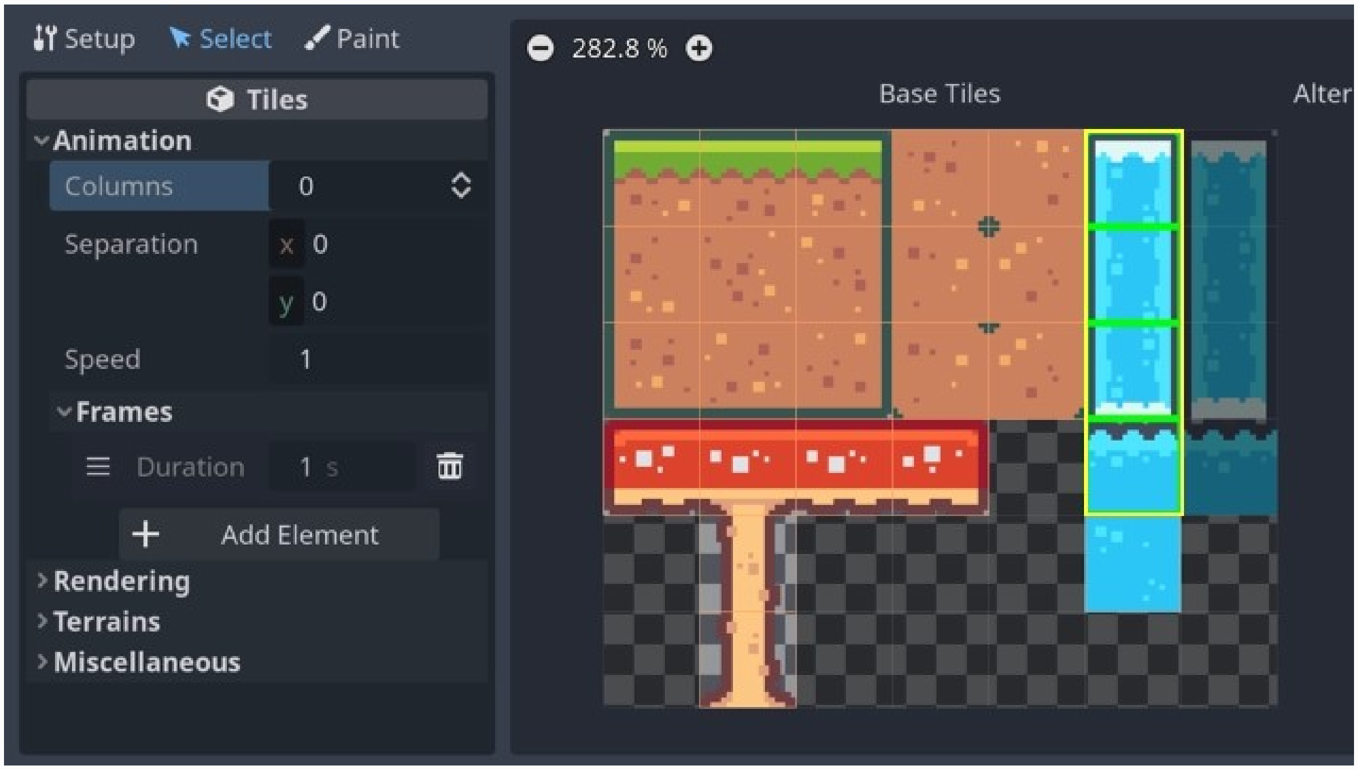
Creating tiles for the first frame of the animation.

Open **Paint** mode and paint the bitmasks for the `Waterfall` and `River` terrains.



Setting up terrain bitmasks.

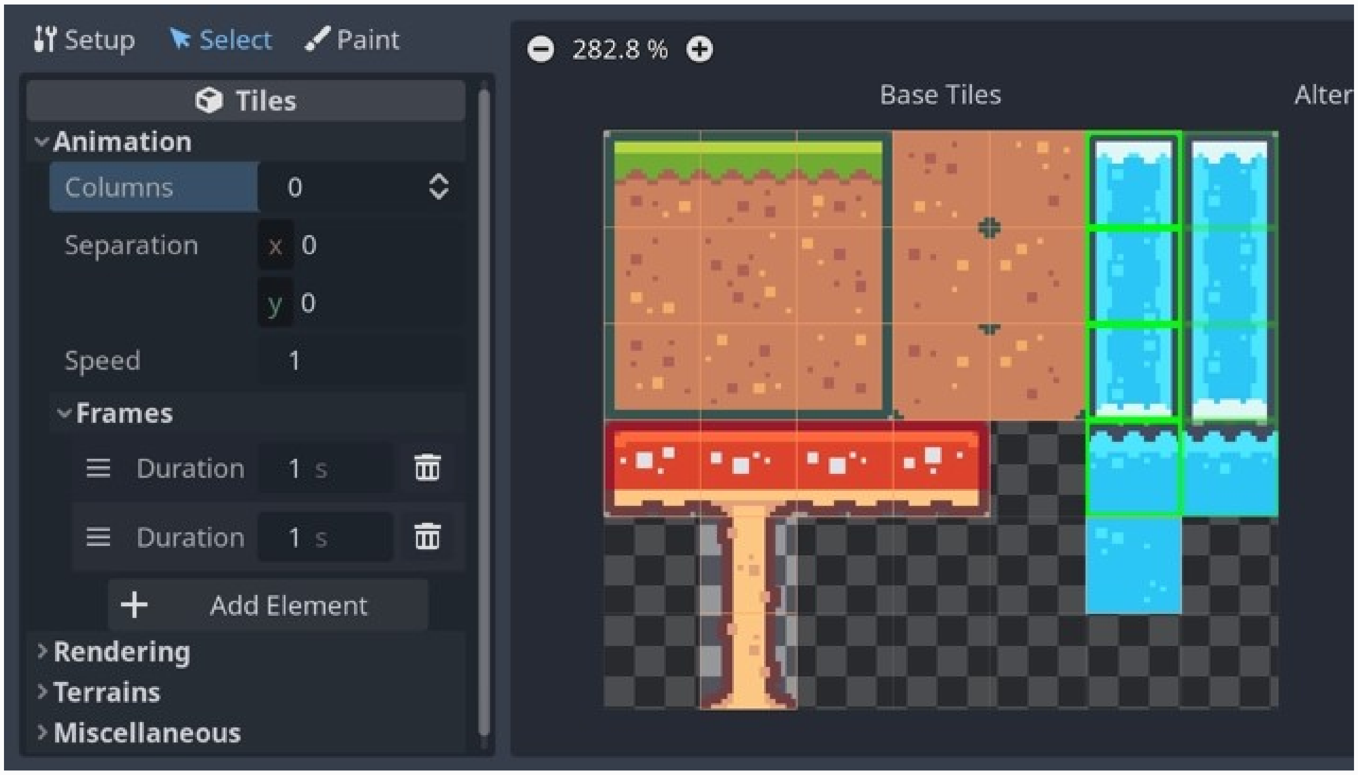
Next, open the **Select** tab and select the tiles in the first frame of the animation. Make sure not to select the bottom river tile, as it does not have an animation. Then expand the **Animation** section.



Animation
Columns 0
Separation x 0 y 0
Speed 1
Frames
Duration 1 s

Selecting tiles to animate.

Under the **Frames** section, press the **Add Element** button to add a new frame.

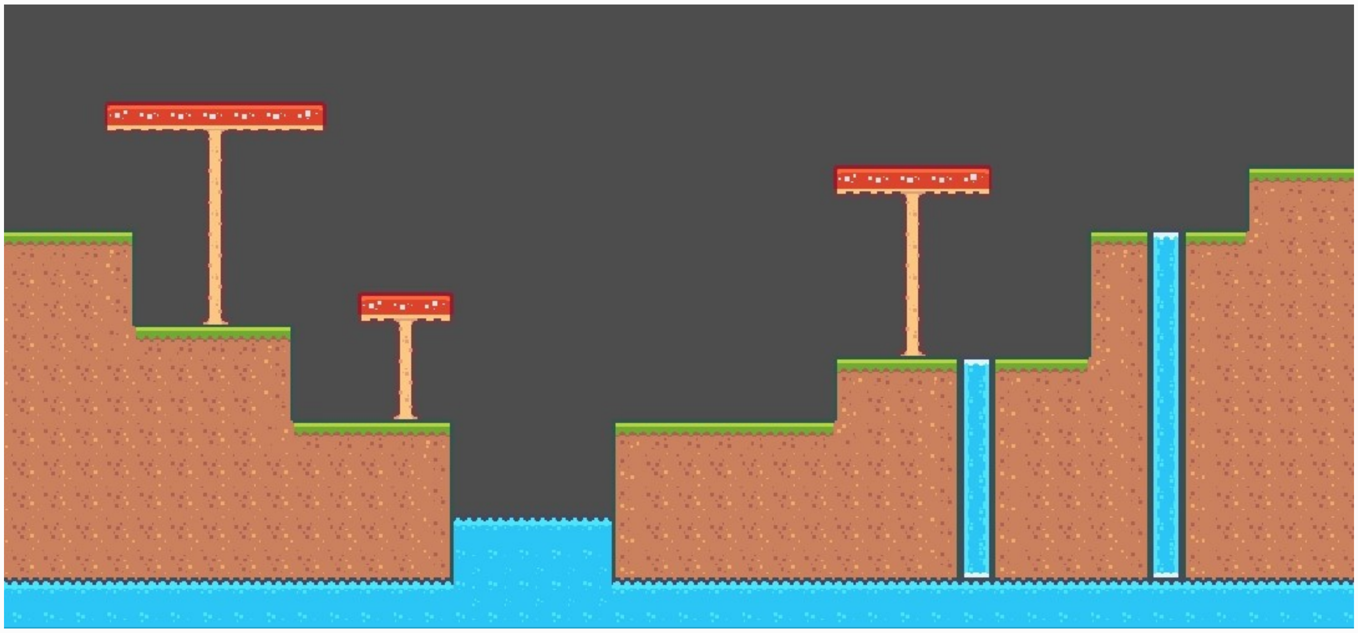


Animation
Columns 0
Separation x 0 y 0
Speed 1
Frames
Duration 1 s
Duration 1 s

Adding a second frame.

The second frame of the animation represents extra textures for the original tiles. They are not their own tiles, so they do not need -- and cannot have -- their own terrain bitmasks or any atlas tile properties.

Now, if we add waterfalls and a river to our TileMap, they will play their two-frame animation in the editor:


Animated TileMap level.

Note

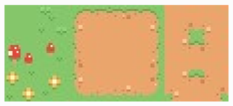
In this example, the river and waterfall tiles were painted on a new TileMap *layer*, so that they are on top of the ground tiles. See Using TileMaps for more information on using layers.

Using probabilities for multiple tiles with the same bitmask

When you have more than one tile with the same bitmask, Godot will choose among them using weighted random **probabilities**. You can use this to add variety to your levels.

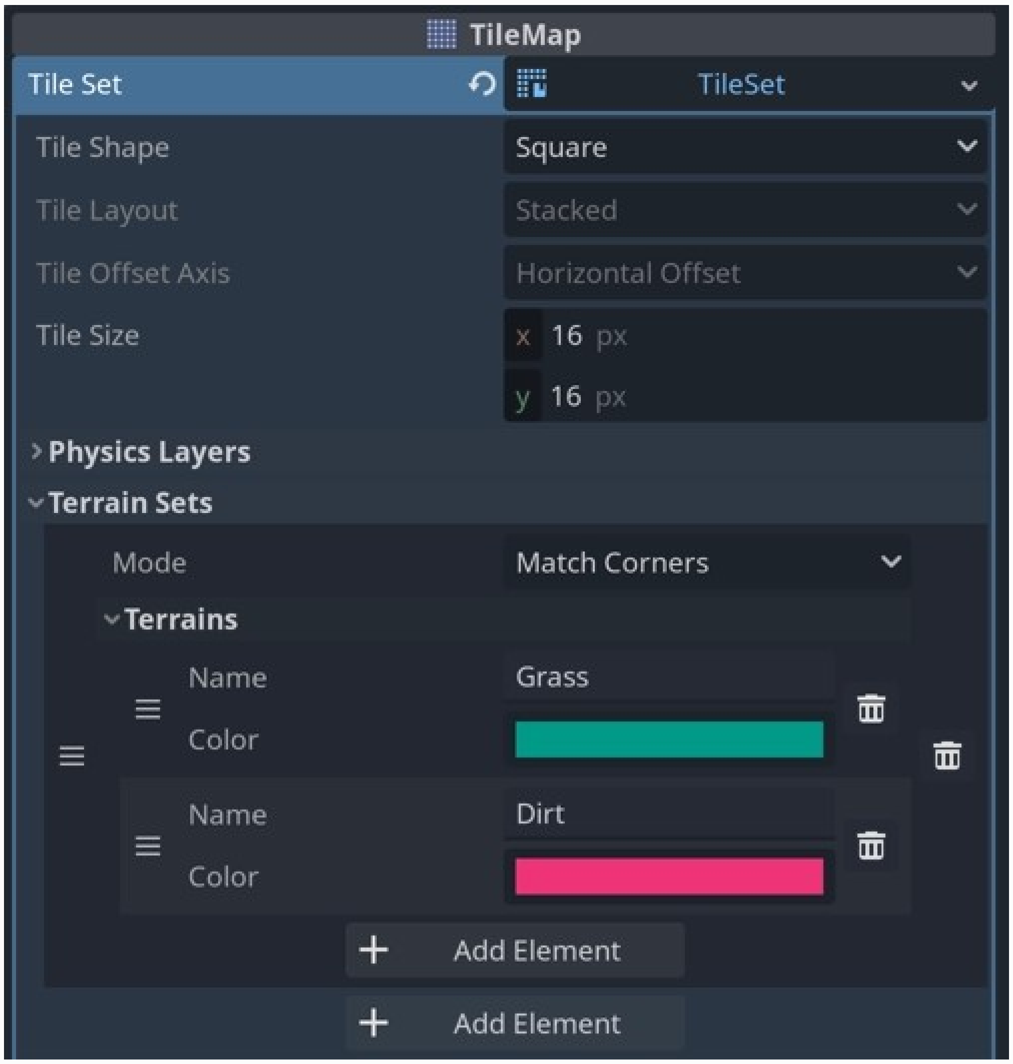
For this example, we will use the Tiny Town tilesheet. It has a plain grass tile at the top left. It also has other variations of it with clovers, mushrooms, and flowers.

Tiny Town tilesheet



*Tilesheet adapted from Tiny Town by Kenney (CC0). Right-click and choose **Save as...** to download.*

First, set up a **terrain set** in Match Corners mode and add `Grass` and `Dirt` terrains.



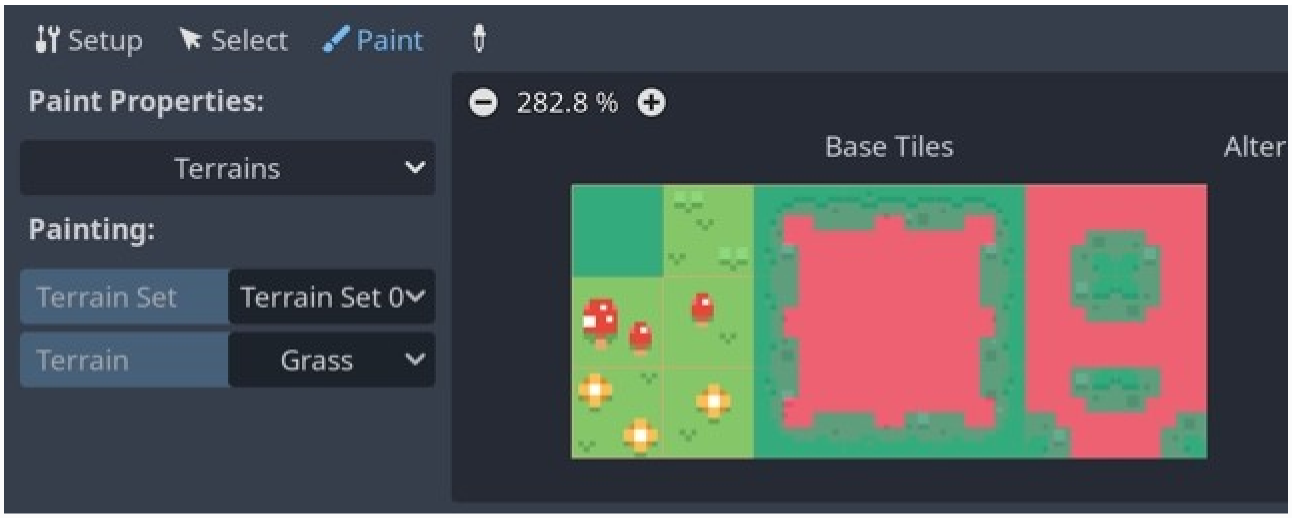
TileMap

Tile Set	TileSet
Tile Shape	Square
Tile Layout	Stacked
Tile Offset Axis	Horizontal Offset
Tile Size	x 16 px
	y 16 px

Mode	Match Corners
Terrains	
Name	Grass
Color	🟩
Name	Dirt
Color	🟥

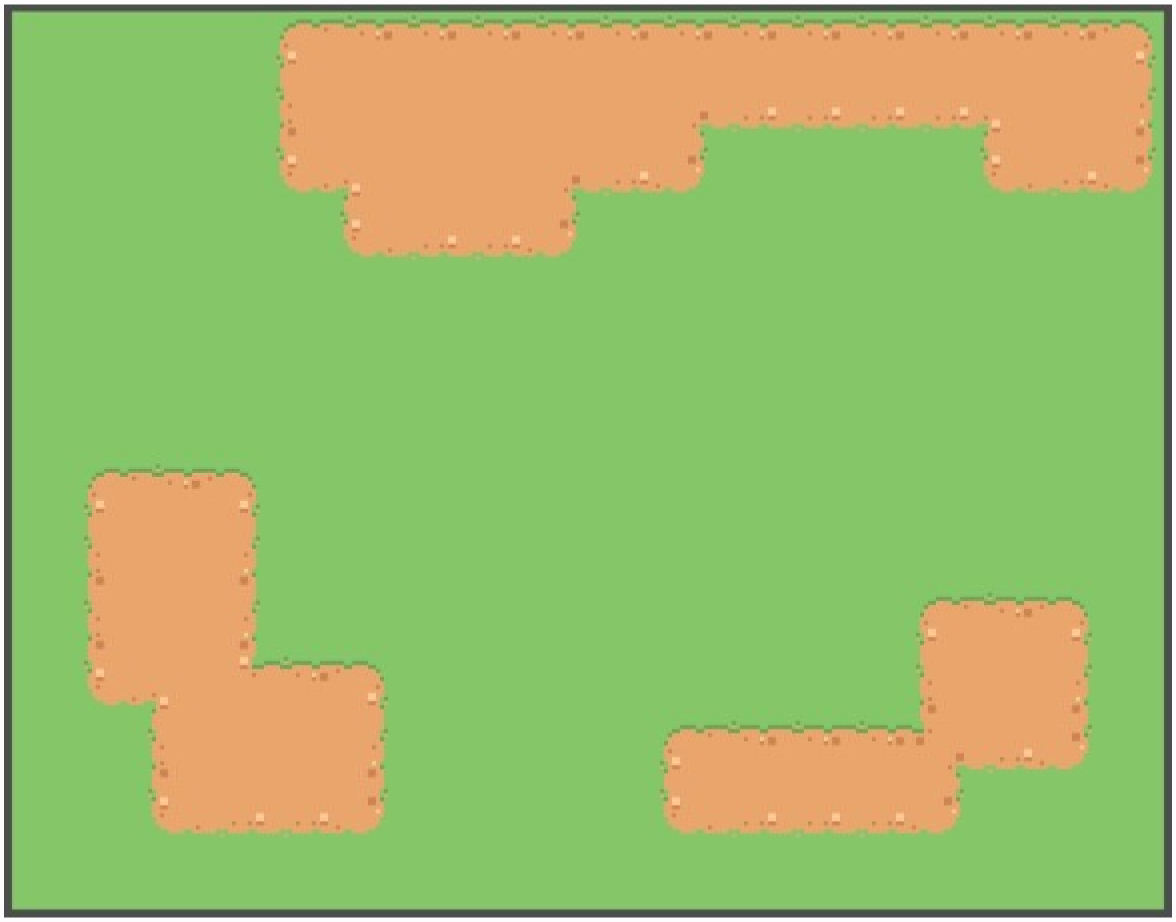
Setting up the terrain set.

In the **TileSet editor**, create tiles and paint the initial bitmask. To start, we will only give the plain grass tile a bitmask.



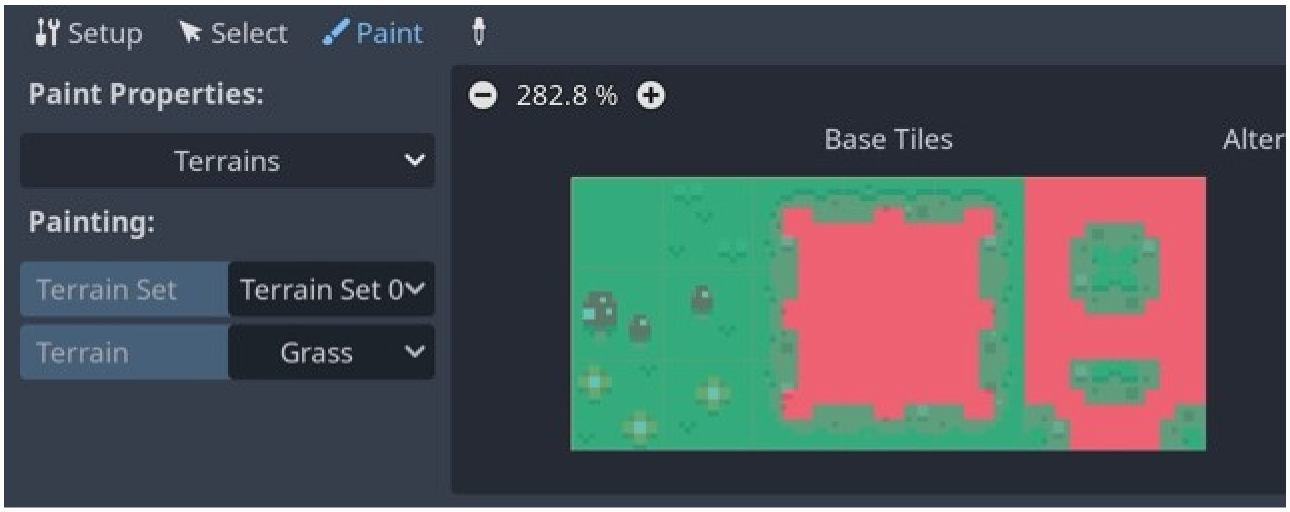
Painting the initial bitmask.

If you paint on the TileMap now, it will look like this:



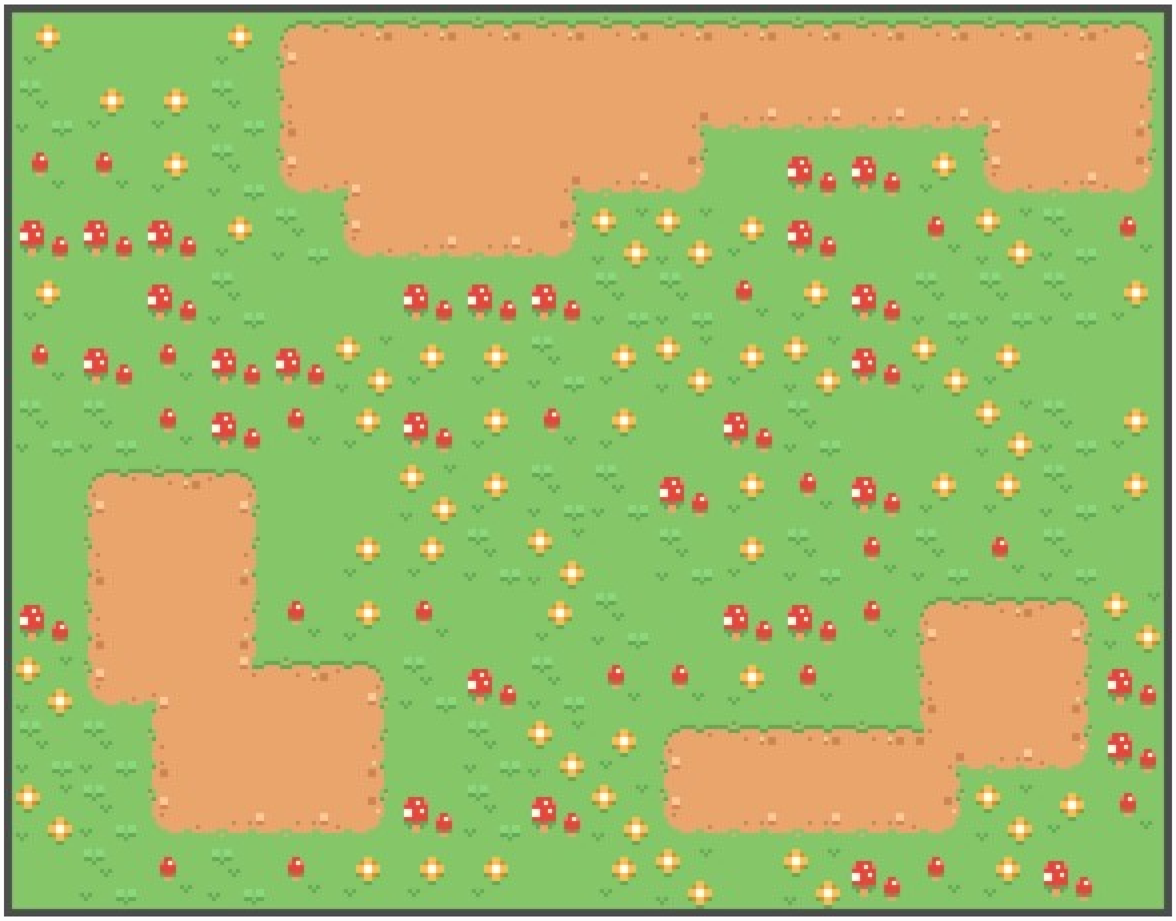
TileMap with a single grass tile.

Now add bitmasks for all the grass tile variations.



Painting bitmasks for grass tile variations.

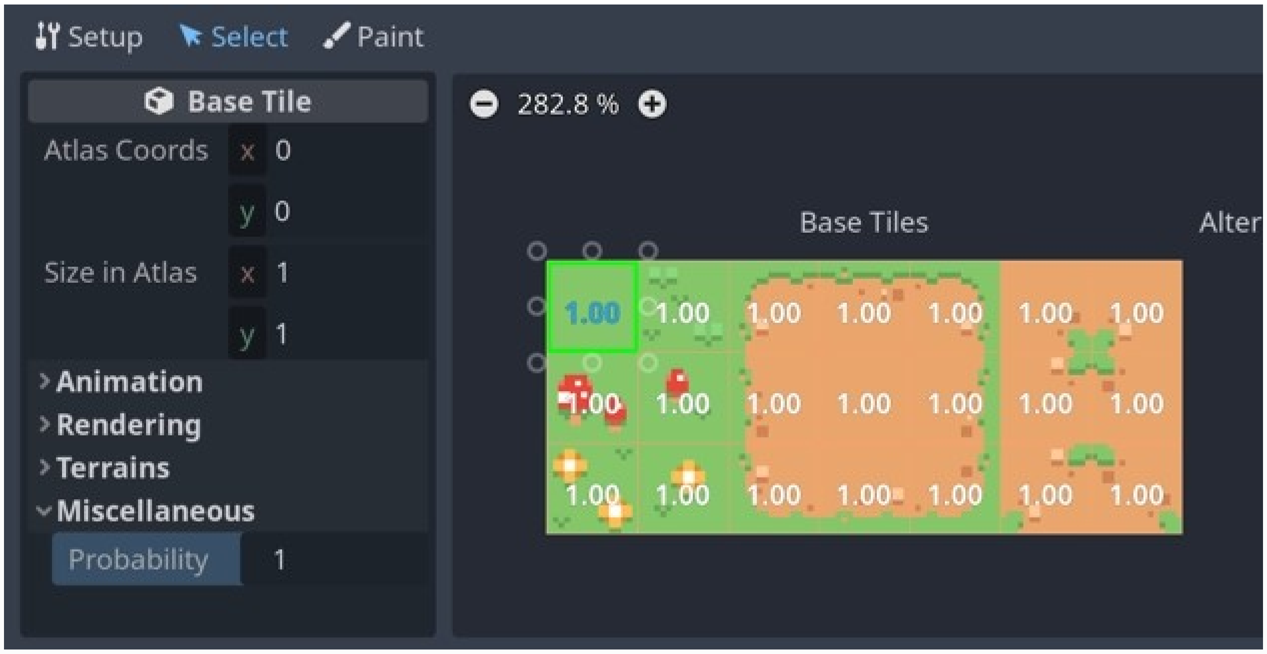
If you re-paint the `Grass` terrain on the TileMap, it will now look similar to this:



TileMap with grass tile variations added.

The TileMap has more variety now, but it is more cluttered than we'd like. Maybe we want the plain grass tile to still be the most common one, and we'd like tiles like the mushrooms to be rare.

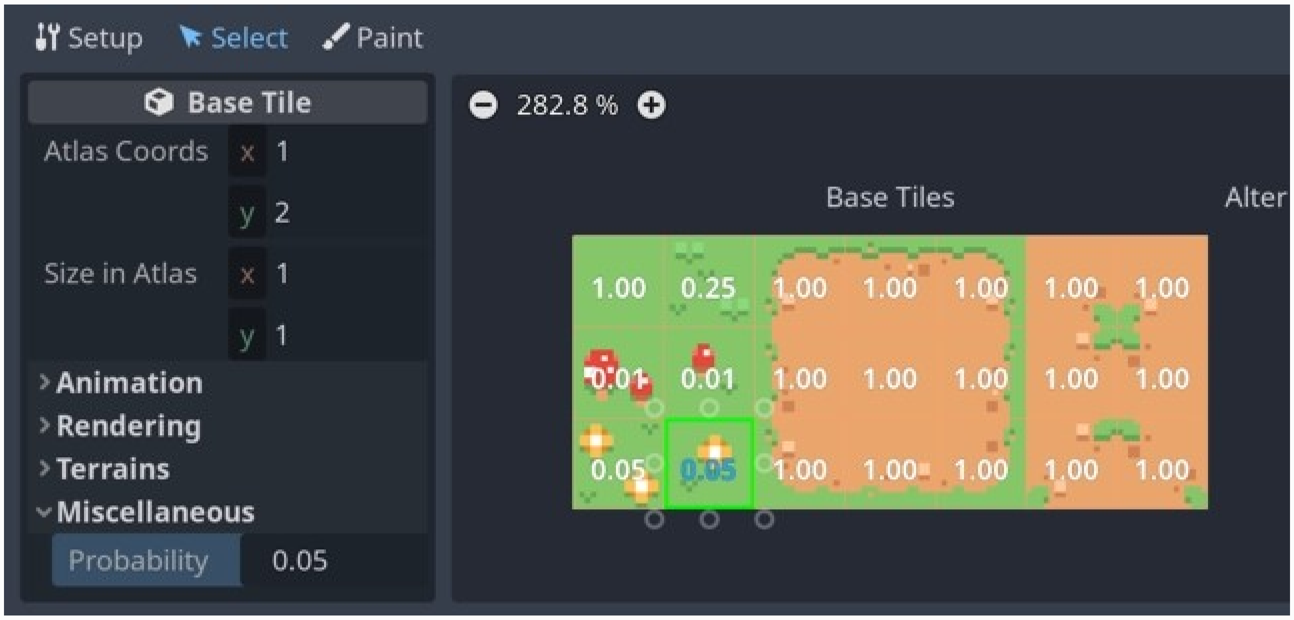
Open **Select** mode and **left-click** on one of the grass tiles to select it. Then open the **Miscellaneous** section. Click on the **Probability** label to see its overlay.



Tiles with default probability values.

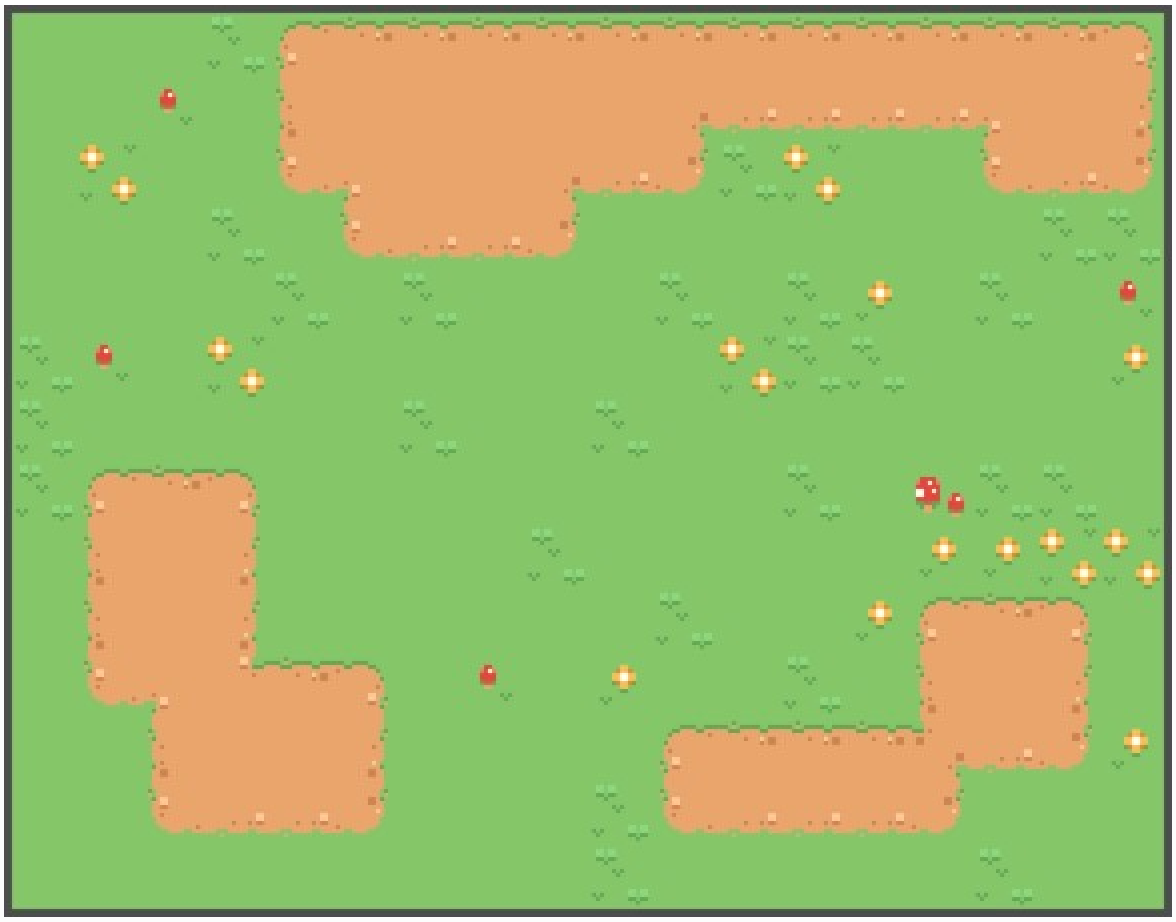
By default, every tile has a probability of `1.00`. For terrain tiles, this value is only relevant when multiple tiles have the same bitmask. So, we will focus only on the grass tiles here.

Try assigning new *probability* values to the grass tiles like this:



Tiles with custom probability values.

If you repaint the `Grass` terrain on the TileMap again, you will now get a result like this:



TileMap level with custom probabilities.

Using alternative tiles for one tile with multiple bitmasks

Note

In Godot 3.x, you could set *ignore bits* in an autotile so that one tile could fill the role of multiple bitmasks. Terrain tiles do not have ignore bits, but you can turn any tile in Godot 4 into an *alternative tile*. If you set different combinations of peering bits on the alternative tiles, they can serve a similar function.

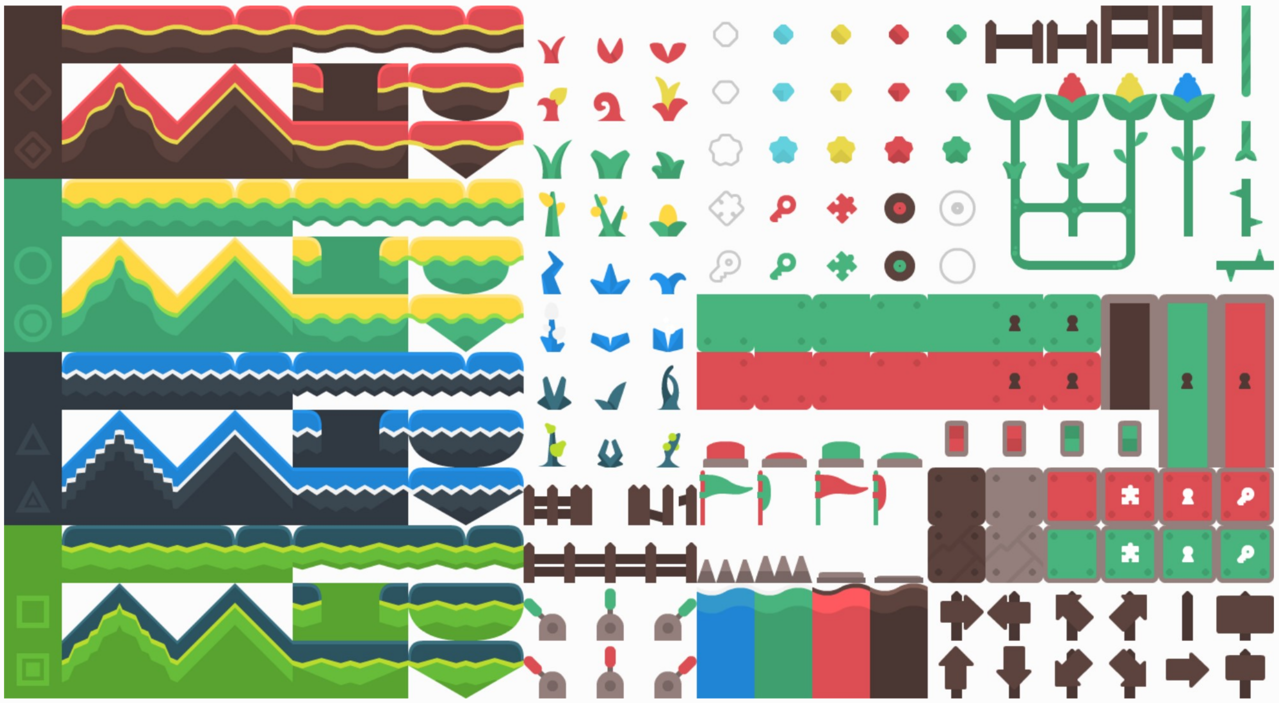
You can use **alternative tiles** to assign different combinations of peering bits to a single tile. One use for alternative tiles is to make a working terrain set from an incomplete set of tiles.

See also

For general information on alternative tiles, see Creating alternative tiles.

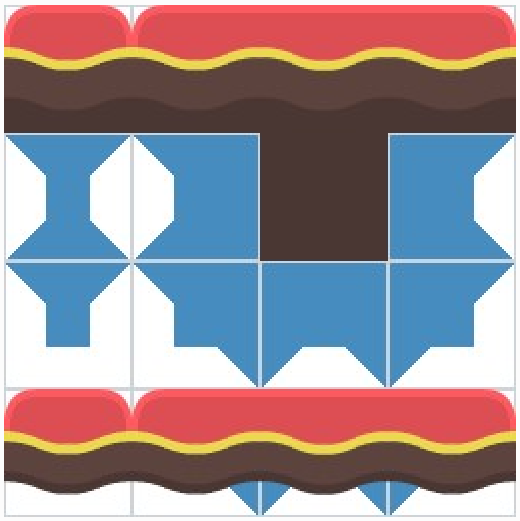
For this example, we will use the Abstract Platformer tilesheet.

Abstract Platformer tilesheet



*Abstract Platformer by Kenney (CC0). Right-click and choose **Save as...** to download.*

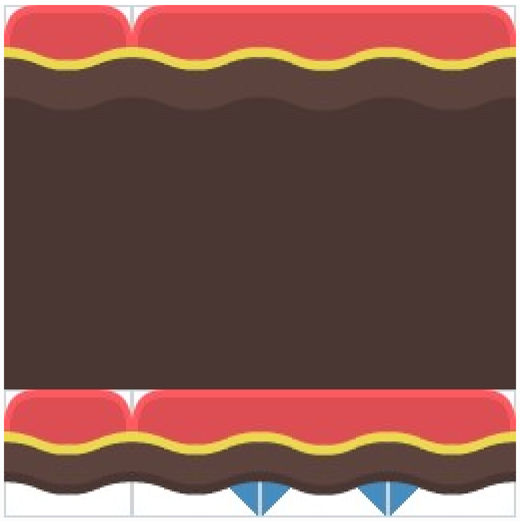
Let's say we want to create a terrain for the first group of tiles at the top. They aren't laid out like a traditional autotile, so we'll use a paint program to arrange them over one of the terrain mode templates to see how they will fit.



Arranging tiles over the template.

The top row of tiles seems like it would work well as a Match Sides terrain set. The only problem is that we are missing some tiles. It seems that the middle tile with the solid color is meant to be reused for the bottom and sides. But we can only assign one bitmask to it. So, if we create the terrain set like this, Godot will have to guess which tiles to use for the other bitmasks, and it may not choose the tile we want.

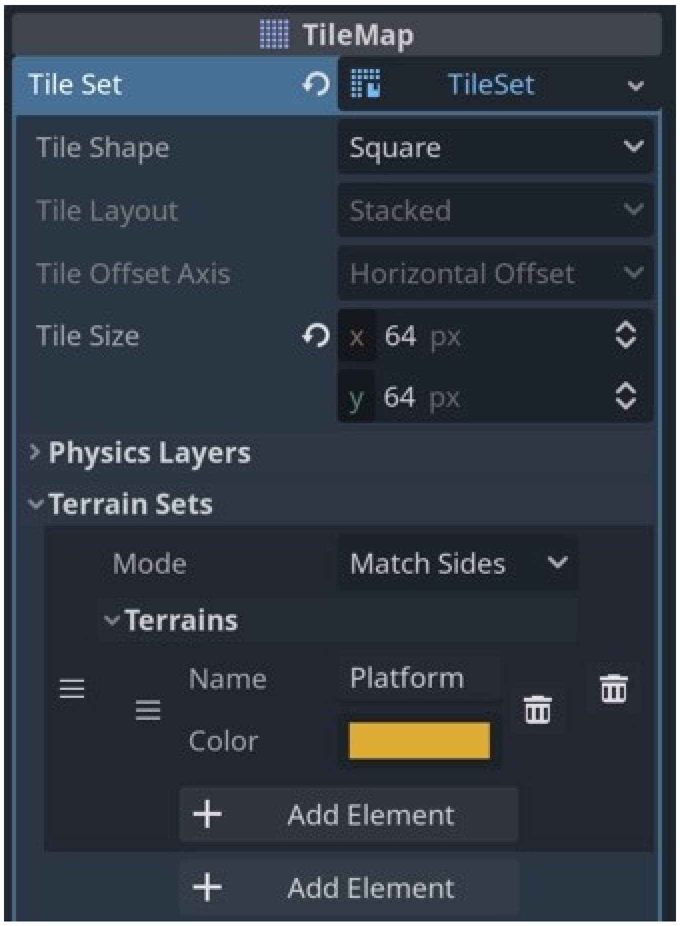
We could create a new tilesheet by duplicating the tile we need in our paint program like this:



Filling in the missing tiles in a paint program.

This would be a good solution. But let's say that we want to use the original tilesheet without creating extra textures. **Alternative tiles** will allow us to reuse the same tile texture for multiple bitmasks.

Let's go back to Godot. Open the **TileSet** inspector and create a terrain set in Match Sides mode. Then add a `Platform` terrain.



TileMap

Tile Set: TileSet

Tile Shape: Square

Tile Layout: Stacked

Tile Offset Axis: Horizontal Offset

Tile Size: x 64 px, y 64 px

Physics Layers

Terrain Sets

Mode: Match Sides

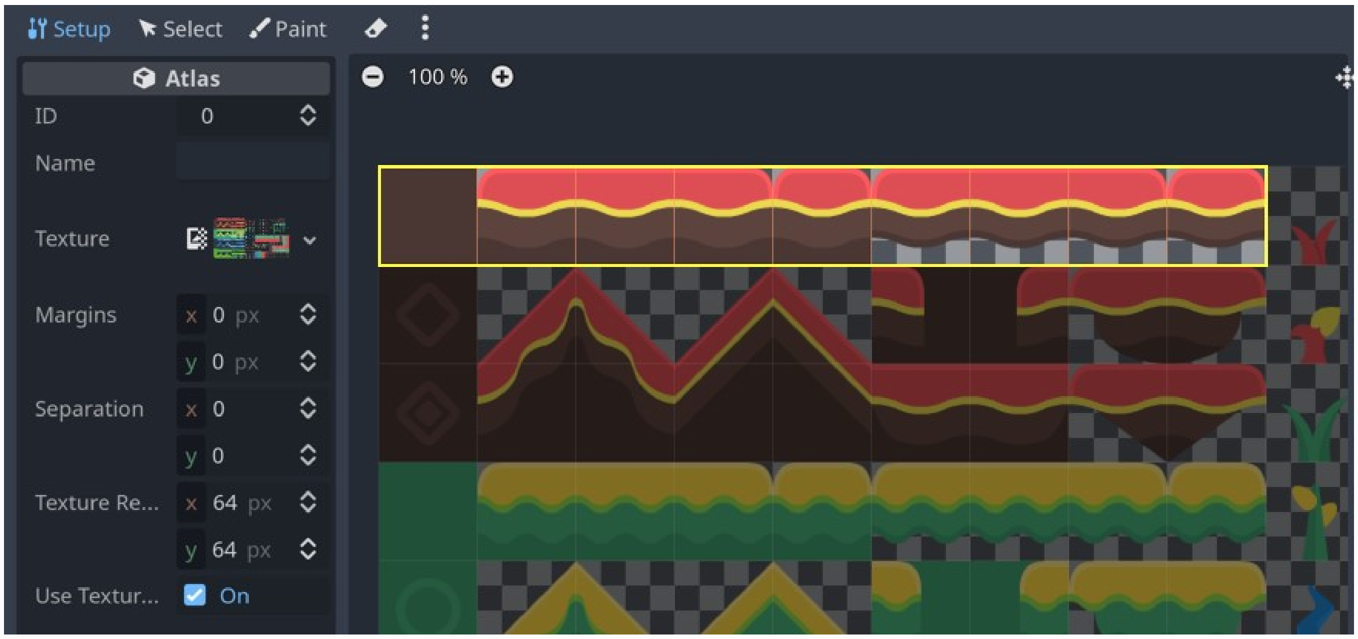
Terrains

Name: Platform

Color: 🟨

Setting up the terrain set.

Then go to the **TileSet editor**, open **Setup** mode, and create tiles for the top row.

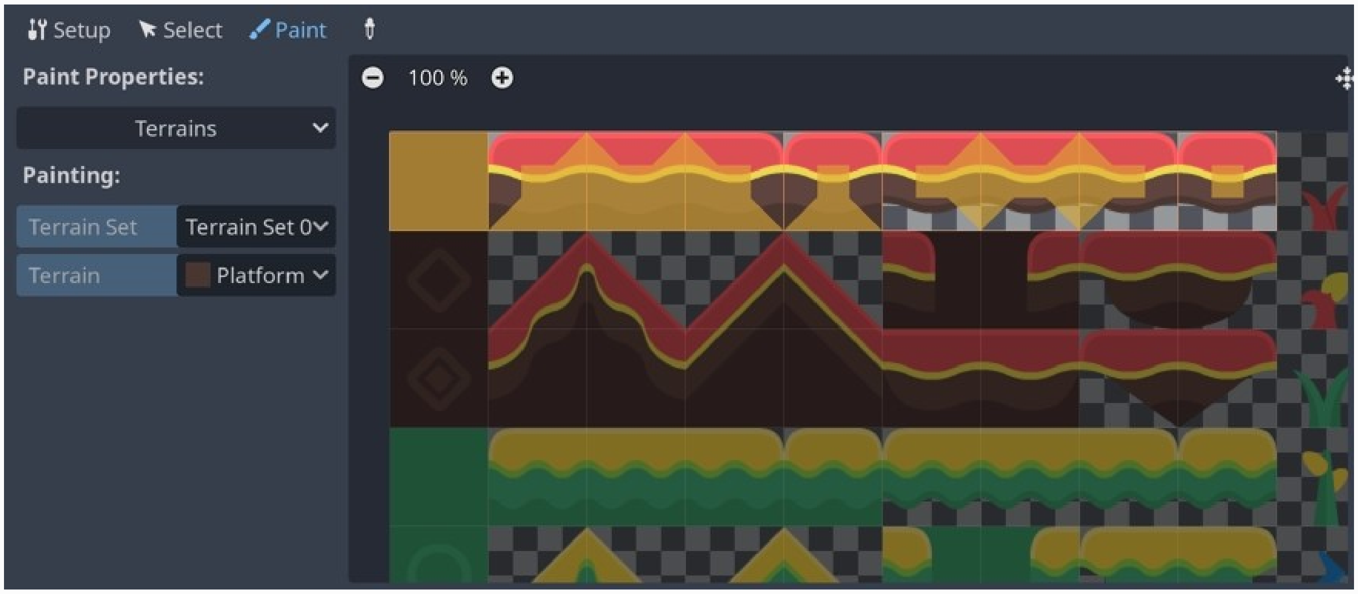


Atlas

ID	0
Name	
Texture	
Margins	x 0 px
	y 0 px
Separation	x 0
	y 0
Texture Re...	x 64 px
	y 64 px
Use Textur...	✓ On

Creating tiles.

Open **Paint** mode and paint the `Platform` terrain onto these base tiles.



Paint Properties:

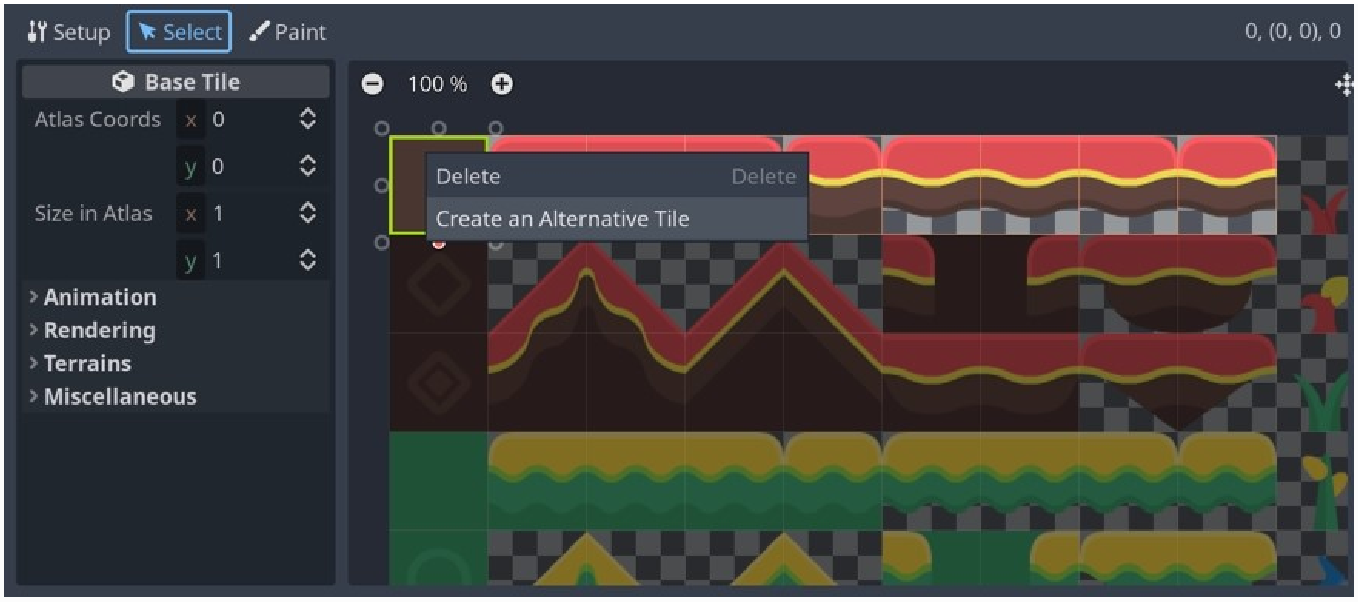
Terrains

Painting:

Terrain Set	Terrain Set 0
Terrain	Platform

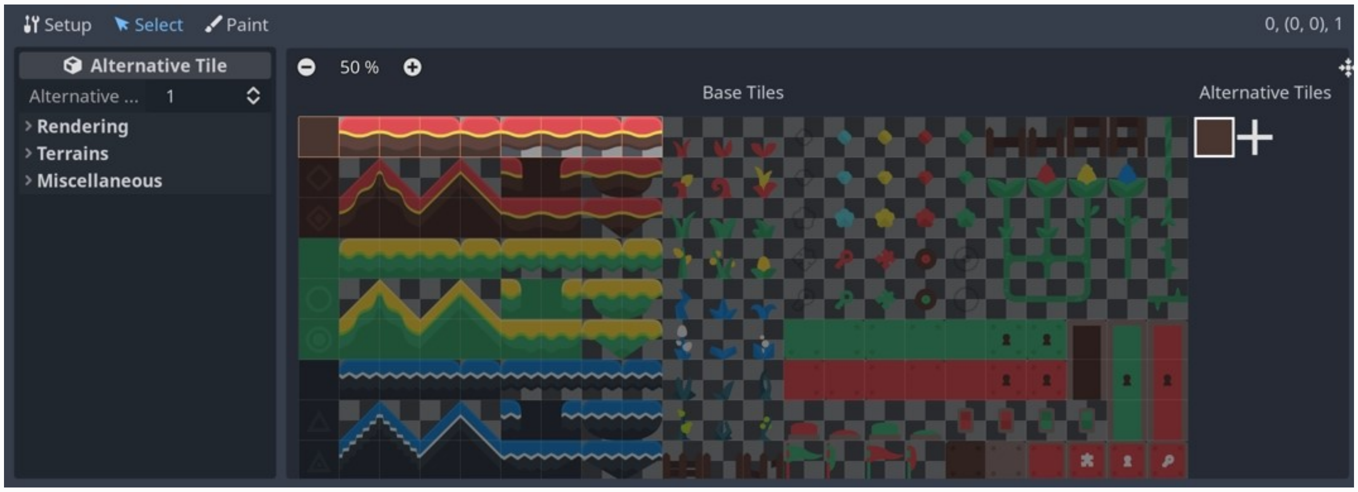
Painting terrains onto the base tiles.

Then open **Select** mode and **right-click** on the first tile at the top-left. In the popup menu, select **Create an Alternative Tile**.



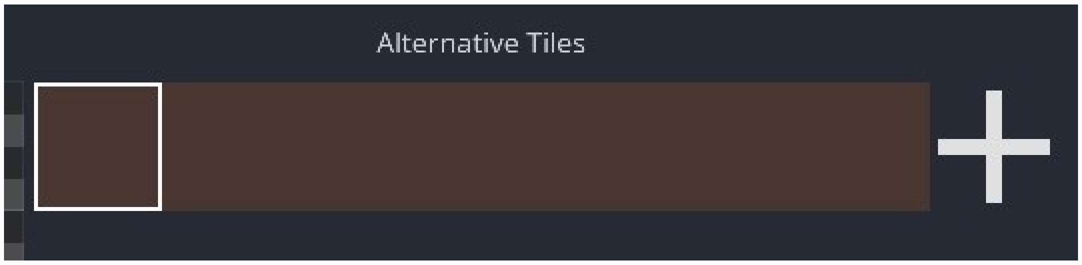
Creating an alternative tile.

The new alternative tile appears to the right of the base tiles. We may need to zoom out to see it.



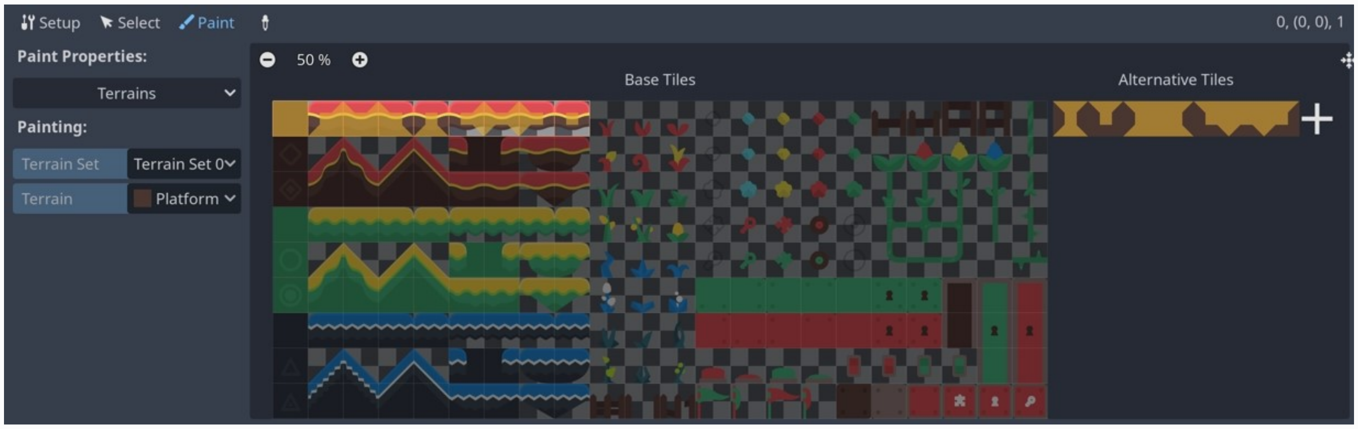
Creating an alternative tile.

Press the `+` button to create more copies of the same tile, until you have the seven that we need.



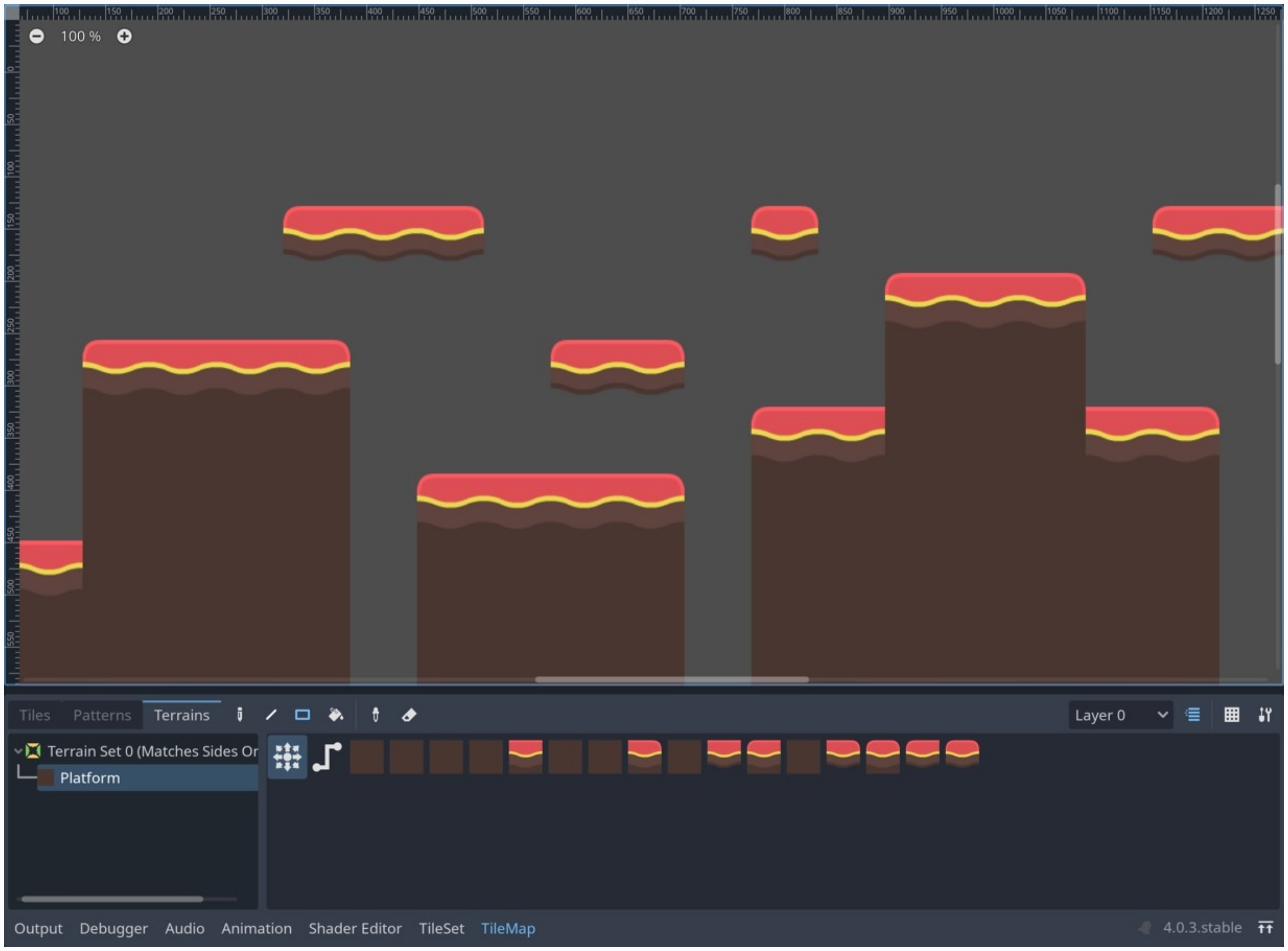
Creating seven alternative tiles.

Go back to **Paint** mode and paint the terrain set and the `Platform` terrain directly onto the new alternative tiles. We can use the Match Sides template from before as a reference for the bitmasks that we need.



Finishing the bitmask.

Now we can switch to the **TileMap editor** and paint a level with our `Platform` terrain.



*In the TileMap **Terrains** tab, the alternative tiles are listed alongside the base tiles.*