Tag Custom Importing Process

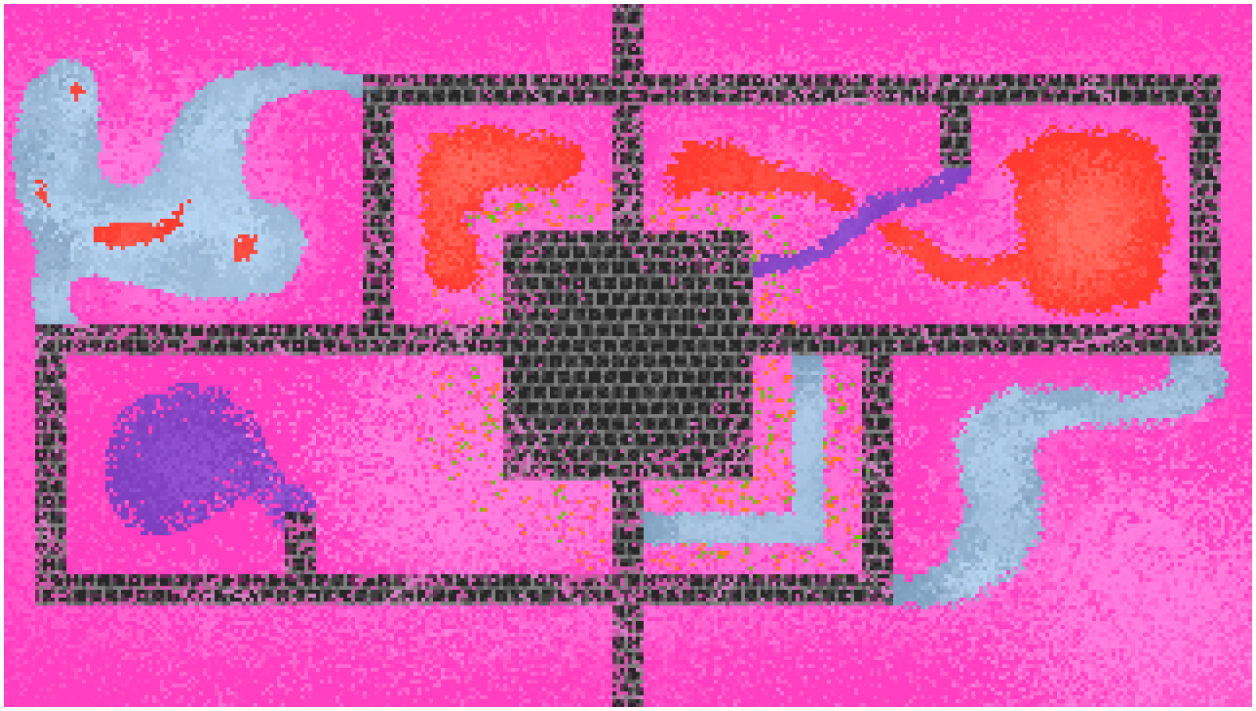
Version 1.0

So, you want to create some custom elements in Tag? Look no further!

The three things that you can customize in Tag are: the map (land), skins, and objects.

The process of creating custom elements is as easy as you let it be. Creating the designs will hopefully be the toughest part, and for the examples below I have created some elements based upon already existing Tag items to demonstrate this process:

My example custom land:



My example custom skins:

My example custom object:

Now let’s take a look into how these items are created.

Creating Custom Maps

To create a custom map, you must initially create an image that is 320 pixels wide and 180 pixels tall (320 x 180 pixels). This may seem small, and yes it is quite small! However, once you've created that image, you must scale it up to 2560 pixels wide, and 1440 pixels tall (2560 x 1440 pixels). Once the image has been scaled, it's all good to go to be imported into tag! If you were to import an image of a different size, it wouldn't crash the game, however the game would look very strange!

Example of an image imported with an incorrect size (this is actually the usual land image file, however it is at its original size (320x180) instead of being scaled up!):

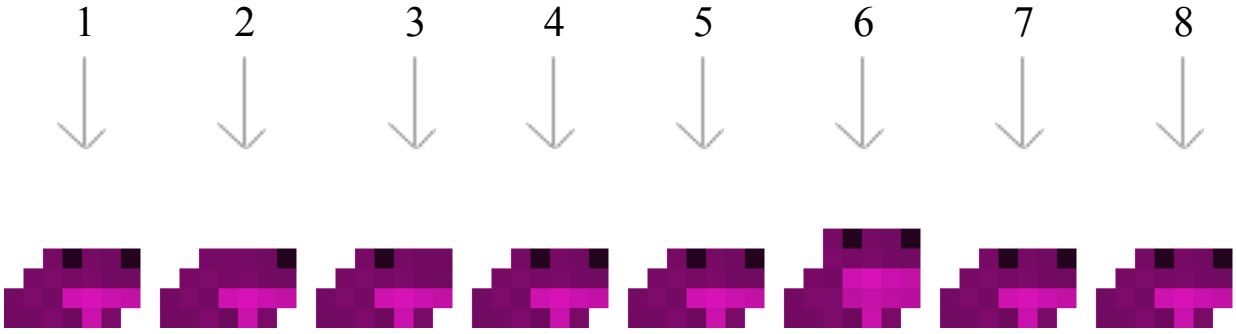
The design of the map is completely up to you! You have complete freedom to fill the space with whatever you'd like, and on top of that, you can create custom objects that match the map's look and feel.

Creating Custom Objects

Creating custom objects is easier and looser than creating custom maps. The size and scale of the object will be determined when you import it, so you can design whatever you'd like in your image file. For this example, I've created an object that is just another color variation on the frogs already in the game:

This image file is 64 pixels wide and 8 pixels tall (64 x 8). If you decide to animate the object you're creating, model it after this image. Each frame of animation comes right after the other, in a straight line, exactly spaced. So for this object, there are 8 frames of animation, and they are each 8 pixels wide and 8 pixels tall (8 x 8):

Frames of Animation:



The frames of animation must be the same size, this is how Tag processes them. As we will see later when we are forming our custom import file, Tag will take in the size of the first (left-most) frame, then it will take in the number of frames of animation, and it will assume that each frame is the same size as the first.

Creating Custom Skins

Now we get to see how we can make custom skins, wahoo! To begin, you must create an image that is 64 pixels TALL, and the width will be determined by how many custom skins you have. In this example:

The image will be 128 pixels WIDE, because there are two custom skins. This means that each skin in Tag is 64 pixels wide and 64 pixels tall (64 x 64). The actual character's size is 16 x 16 pixels, however we have 4 frames of animation for each row / mode, so this comes out to be 4 rows of 64 pixels wide. The rows / modes are as follows:

Row / Mode 1: Not moving
Row / Mode 2: Moving
Row / Mode 3: First emote
Row / Mode 4: Second emote

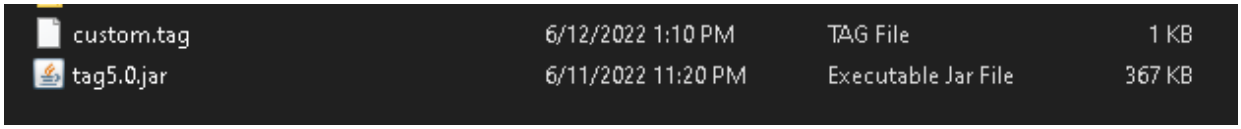
Similar to creating a custom object, the frames of animation for a given mode come right after one another in a straight line, in little 16 x 16 pixel chunks. The first frame of animation is the left-most 16 x 16 chunk.

Forming the Custom Import File

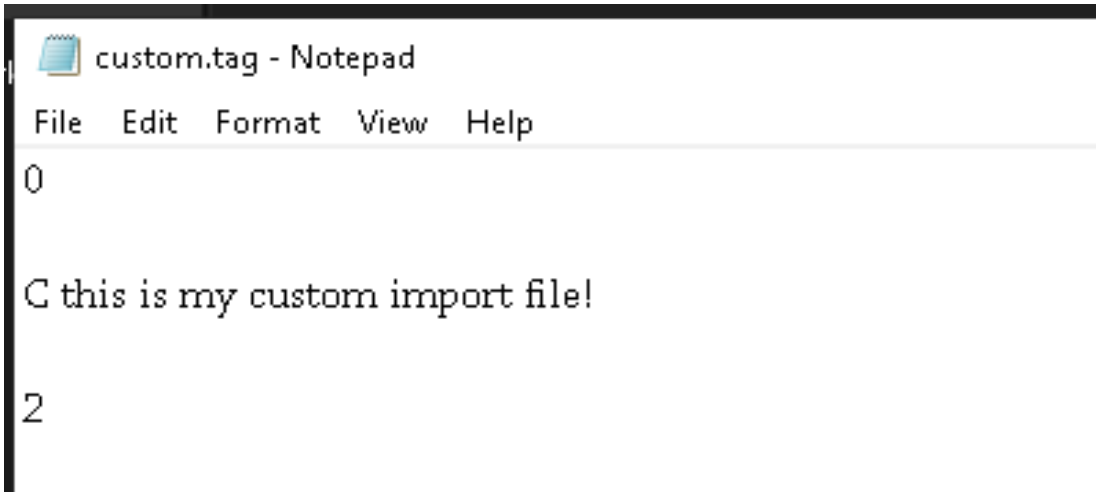
Now to actually import the custom elements! We will take this step by step, first importing the map, then the object, then the custom skins, all from the above examples.

To begin, here are some important notes about the custom import file:

- The file is called custom.tag
- custom.tag is placed in the same directory / folder as the Tag game:



- Once the custom import file has been found, all normal objects will NOT be created, so your map may look pretty sparse if you don't add some objects of your own!
- IMPORTANT - The file must start with a number indicating the version of custom importing, and it must end with the number 2. So, considering this is version 1 of custom importing, I have used 0 as the current version (often times in programming 0 represents the first of something), so always start your custom file with 0, and end it with 2:



If you wish to add comments to your custom file, do so by beginning the line with a capital c, like the above example. The custom import process will ignore this line.

Adding A Custom Map

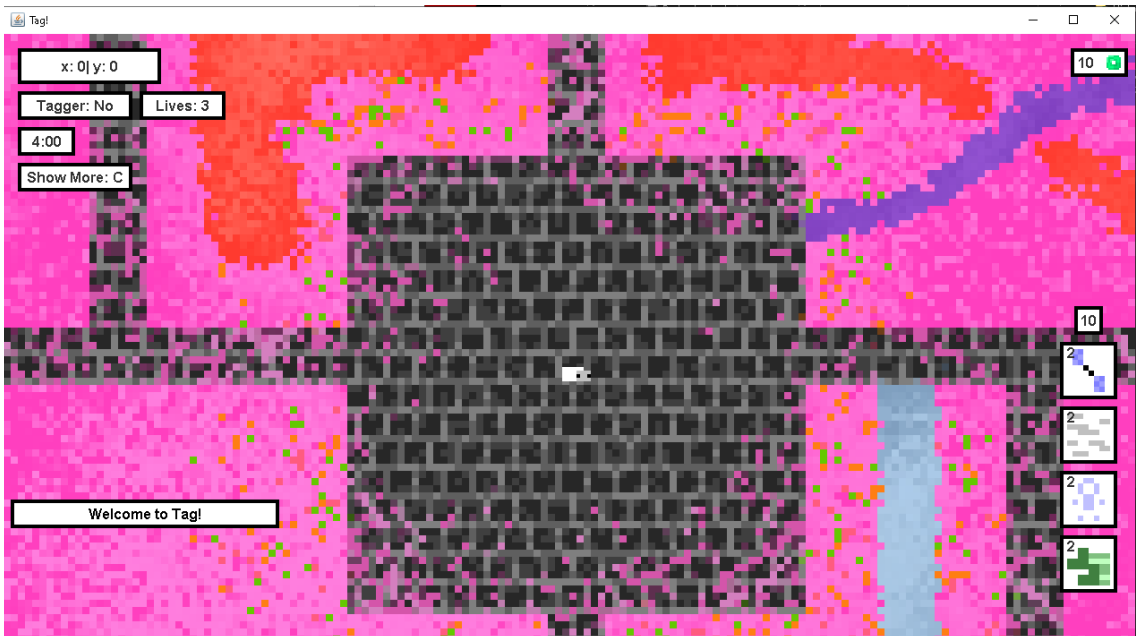
Let's add in our custom map! To do this, you begin a line with "fl" (stands for File for Land), and then you add in the path to where that file is. For example, I have my example land file in C:\p\Java\tag, so I will specify its path as C:\p\Java\tag\customLand.png (the name of the custom land file is customLand.png):

custom.tag - Notepad

File Edit Format View Help

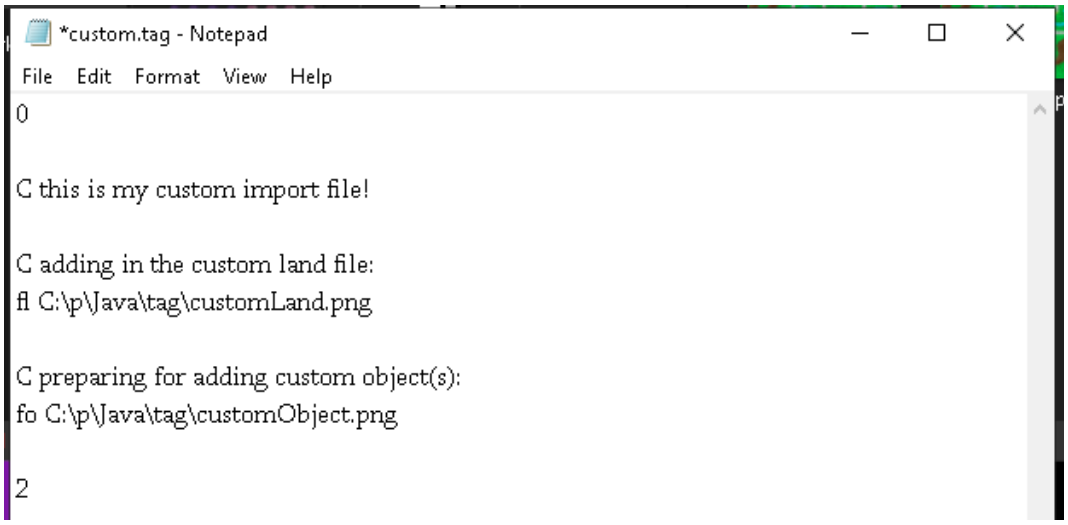
```
0

C this is my custom import file!

C adding in the custom land file:
fl C:\p\Java\tag\customLand.png

2
```

If we were to save this file and open Tag, it should look like this:



Adding A Custom Object

Adding a custom object is the most involved step. Before you add any custom objects, you must specify the file that you will use for all of the custom objects, and this is done similarly to adding a custom map. You start a line with “fo” (stands for File for Objects), and then you specify the path to where that file is. For this example, the frog image from above is at C:\p\Java\tag\customObject.png:



Now that we’ve specified the file that we get the image data from, let’s define our custom object. The things that you must specify for your custom object are:

- The object’s in-game top-left corner x position
- The object’s in-game top-left corner y position
- The object’s in-game width
- The object’s in-game height
- The first pixel on the left side of the first frame of animation
- The first pixel on the top side of the first frame of animation
- The width of a frame of animation
- The height of a frame of animation
- The number of frames of animation (can be 1 if it’s not animated)
- If the object is always on top, or if you can stand in front of it

That's a lot of information, but let's form an object together to get a better idea of how its done. Let's say we want a really big frog to be near the starting point! We can place it a little to the left and a little ways up. To begin forming the frog's data, we start a line with "co" (stands for Custom Object):

```
C preparing for adding custom object(s):
fo C:\p\Java\tag\customObject.png

C this is the frog right here!
co
```

Now that we've started the line, let's tell Tag where he is going to go!

The coordinate system for Tag is just like a normal mathematical graph: moving right increases x, and moving up increases y. So, if we want the frog to be a little to the left and a little bit up, we will have an x value lower than 0 and a y value higher than 0 (because the starting point is x=0, y=0). Let's say for demonstration purposes that we want his coordinates to be x=-140, y=140. Let's add these values to the custom object line:

```
C preparing for adding custom object(s):
fo C:\p\Java\tag\customObject.png

C this is the frog right here!
co -140 140
```

Now let's tell Tag how big it will be! Let's say it's going to be 128 pixels by 128 pixels:

```
C preparing for adding custom object(s):
fo C:\p\Java\tag\customObject.png

C this is the frog right here!
co -140 140 128 128

2
```

Great, one portion done! Now we need to tell Tag where in the image file it is. Let's take another look at the example file:

So, the first frame of animation is at the very left of the image, and we know from before that this image is 64 x 8 pixels.
To tell Tag where the first frame is, we tell it the top left corner of the first frame. The first frame is right at the beginning of the file, so its actually at (0,0), and so that is what we'll tell Tag:

```
C this is the frog right here!
co -140 140 128 128 0 0

2
```

Then we need to tell Tag how big the first frame is. In this example image, each frame is 8 x 8 pixels:

```
C this is the frog right here!
co -140 140 128 128 0 0 8 8

2
```

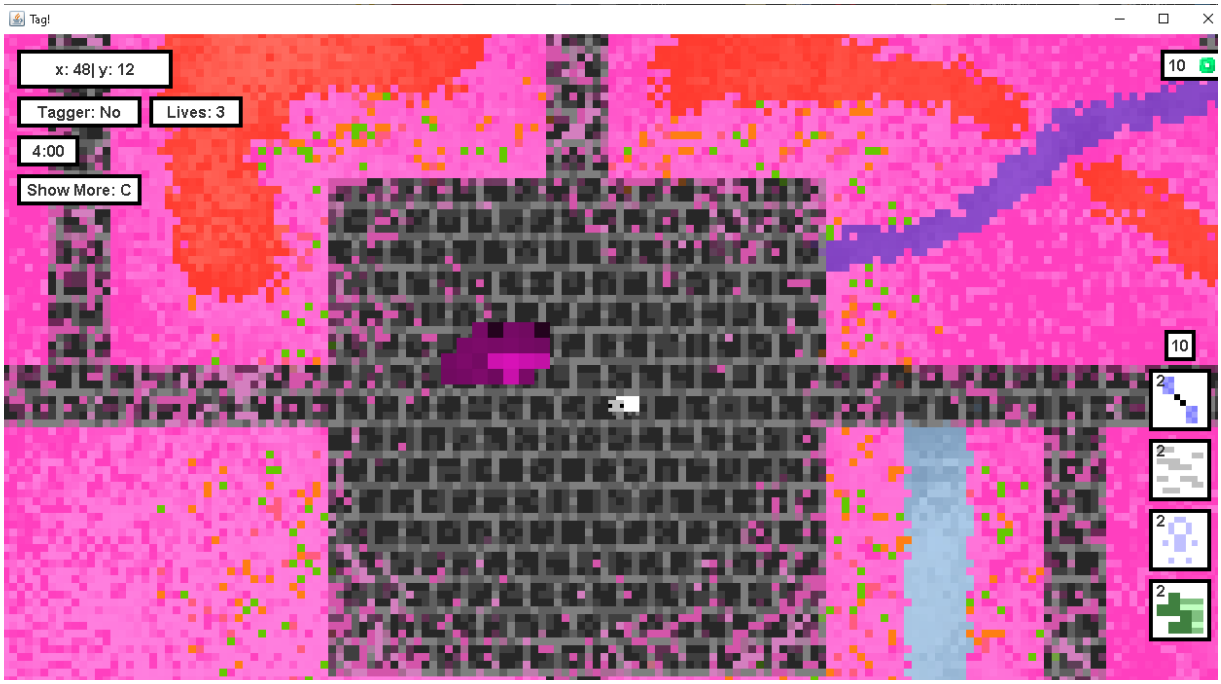
Now let's tell Tag how many frames of animation there are. In this example image, there are 8 frames:

```
C this is the frog right here!
co -140 140 128 128 0 0 8 8 8
```

Now we need to tell Tag if this object is above the players at all times, or if we can stand in front of it. If this object is always on top, add 'T' to the line. If the object can be walked in front of, add 'F'. We want to be able to stand in front of our frog, so we'll add 'F':

```
C this is the frog right here!
co -140 140 128 128 0 0 8 8 8 F
```

That's it, you've created your custom object! Let's save the file and see what it looks like:

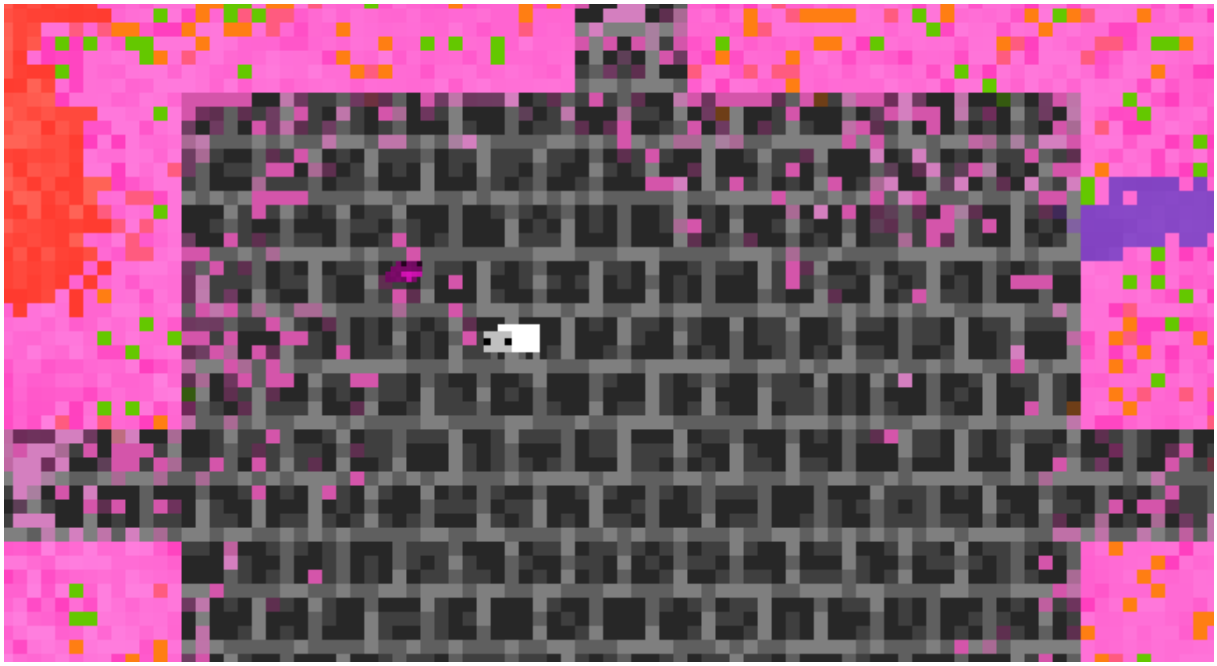


Now that's a big frog.

Let's say we decided it was too big, and we wanted him to be tiny instead. We can simply go back into custom.tag and change his size:

```
C this is the frog right here!
co -140 140 24 24 0 0 8 8 8 F
```

Notice how we changed his width and height from 128 pixels to 24. So now it should be small compared to our character:

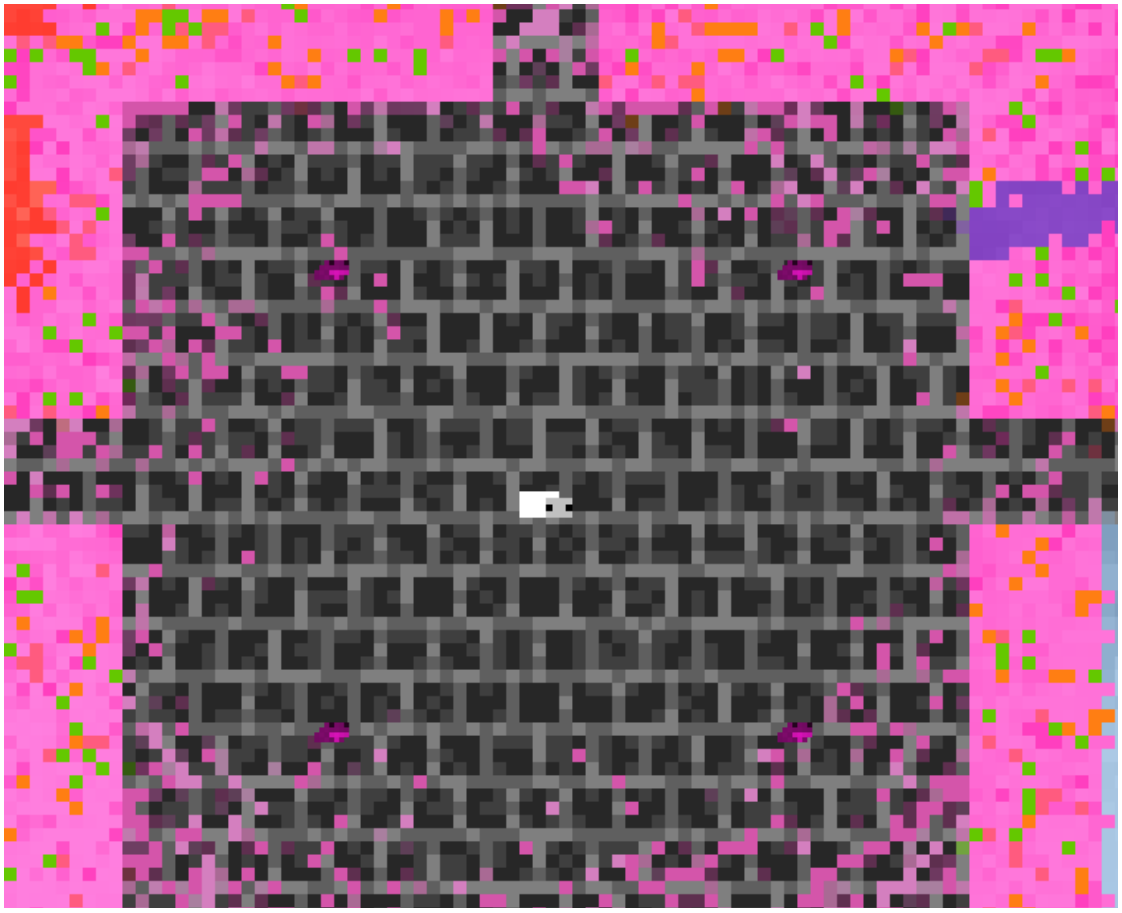


Now that's a tiny frog!

Once you've specified the custom object file, you can make as many custom objects as you'd like from it. I'm going to add in a few more tiny frogs around the starting point:

```
C here come the frogs!
co -140 140 24 24 0 0 8 8 8 F
co 140 140 24 24 0 0 8 8 8 F
co -140 -140 24 24 0 0 8 8 8 F
co 140 -140 24 24 0 0 8 8 8 F
```

And here's what it should look like:



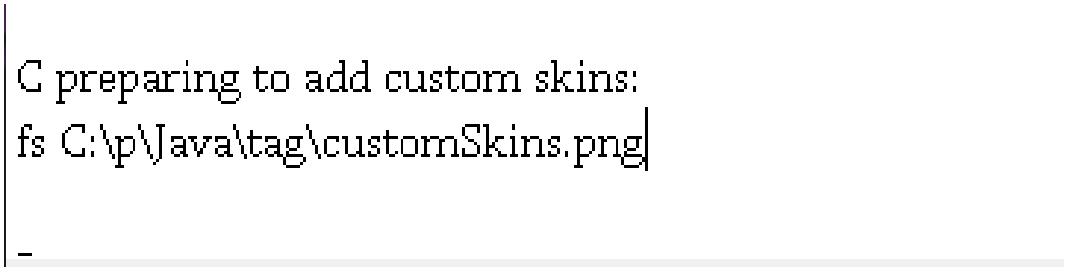
You can barely see them because they blend in so much! We can always change this if we resize them, relocate them, or edit the image file to make them a different color.

Adding A Custom Skin

In this example, we will actually add two custom skins, but we will add them one at a time so the process is clear.

Before you add a custom skin, you have to specify the image file that contains the custom skins. This is done by starting a line with “fs”

(stands for File for Skins), and then typing the path to the custom skin image file, similarly to how we specified the custom land and object files:



To add a custom skin, you must start a line with “cs” (stands for Custom Skin), then you specify two things:

- The index of the skin you wish to change (beginning at 0)
- The index of the custom skin in the image file (also beginning at 0)

To better visualize adding a custom skin, here is the image file we’ll use for custom skins:

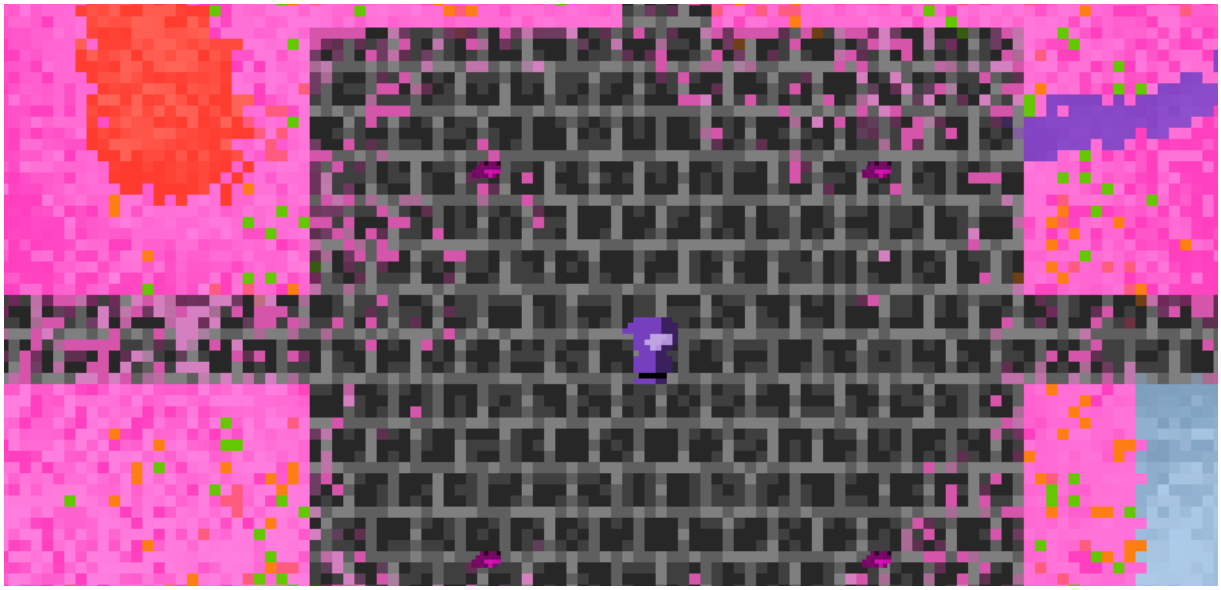
Remember, each custom skin is 64 x 64 pixels.

So, let's say we want to replace the very first skin (the white sheep) with our first custom skin in our file. This is done like so:

```
C here's the first custom skin!
cs 0 0
```

Remember, the first skin of either list (whether its your custom skins or the skins in-game) is known as 0, and it goes up from there. So the second skin would be 1, third would be 2, etc.

Here's what it would look like if we saved our custom file and opened Tag:

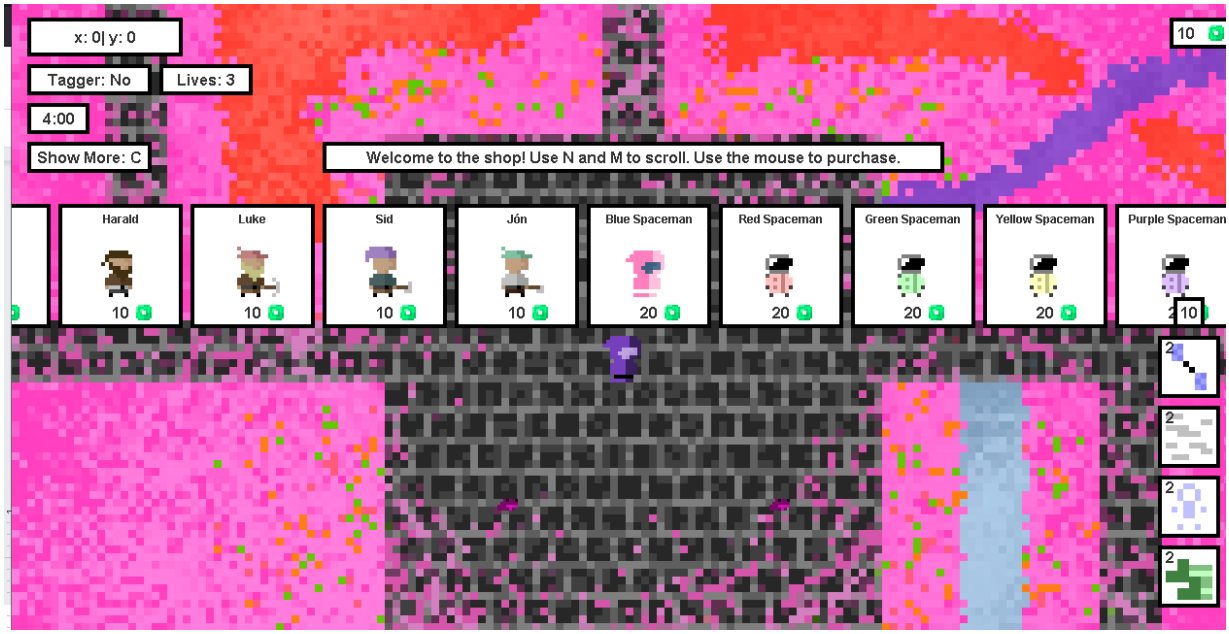


Now our normal sheep outfit has been changed to the first custom skin!

Let's say we want to change one of our spaceman outfits to be the second custom skin. Note that the first skin we can purchase in the shop (as of right now, is "Trom") is at index 4, because it is the 5th skin overall (the four sheep are the first 4). The index for our first spaceman is 13, the blue spaceman. Let's add in a line to replace this skin:

```
C here's the first custom skin!
cs 0 0
C replacing the blue spaceman
cs 13 1
```

So, we've said to replace index 13 with our custom index 1. Let's see this in action:



Notice how even in the shop the skin has been replaced!

Final Notes

I hope this guide has been helpful for explaining custom importing for Tag. There are countless things you can do with this, and hopefully as certain yearly events roll around I can develop custom packs, such as a Halloween pack, Christmas pack, etc. Thank you so much for reading through, I hope that you enjoy Tag!

Jack Lindsay (Vedasys)