Crafting Recipes

Posted on June 29, 2018

Now that we have plenty of items in our mod, we need to add crafting recipes so we can obtain these items in survival. As of 1.12, crafting recipes are now added using JSON files.

Let's make a crafting recipe for our basic item. The crafting recipe is represented using three strings of three characters each. This forms the 3 x 3 crafting grid. If a recipe takes up less space than that 3 x 3 grid, it will automatically work in any other compatible placement, including the 2 x 2 inventory crafting grid if applicable. For example, here's the torch recipe JSON:

```
{
  "type": "minecraft:crafting_shaped",
  "pattern": [
    "X",
    "#"
  ],
  "key": {
    "#": {
      "item": "minecraft:stick"
    },
    "X": [
      {
        "item": "minecraft:coal",
        "data": 0
      },
      {
        "item": "minecraft:coal",
        "data": 1
      }
    ]
  },
  "result": {
    "item": "minecraft:torch",
    "count": 4
  }
}
```

There are a few things to note here. First, we see the `type` tag, defining the type of recipe. This one is a shaped crafting recipe.

Next is the `pattern` tag, which defines the pattern in the crafting grid. Note that the recipe itself only takes up a 1 x 2 area, and so it is defined as such. The recipe will work in any place in the crafting grid, as long as it has the same shape.

After that, each item/block is represented by a character, which is defined in the `key` section. This should be pretty self-explanatory; the `item` tag corresponds to the registry name of an item or a block. Also, you can define metadata afterwards using the `data` tag. The torch uses this to allow coal or charcoal to be used, as charcoal is the same item as coal with different metadata. Also note how this recipe maps two items to one character, so they can be used interchangeably. The wildcard data value `32767` could also have been used.

Finally, in the `result` tag, we find the item resulting from this recipe, as well as the count (optional). The `data` tag can also be used here.

Now that we've covered shaped recipes, let's go over shapeless recipes. These are much simpler, because they don't require a pattern and can just list the required items. Magma cream is a good example:

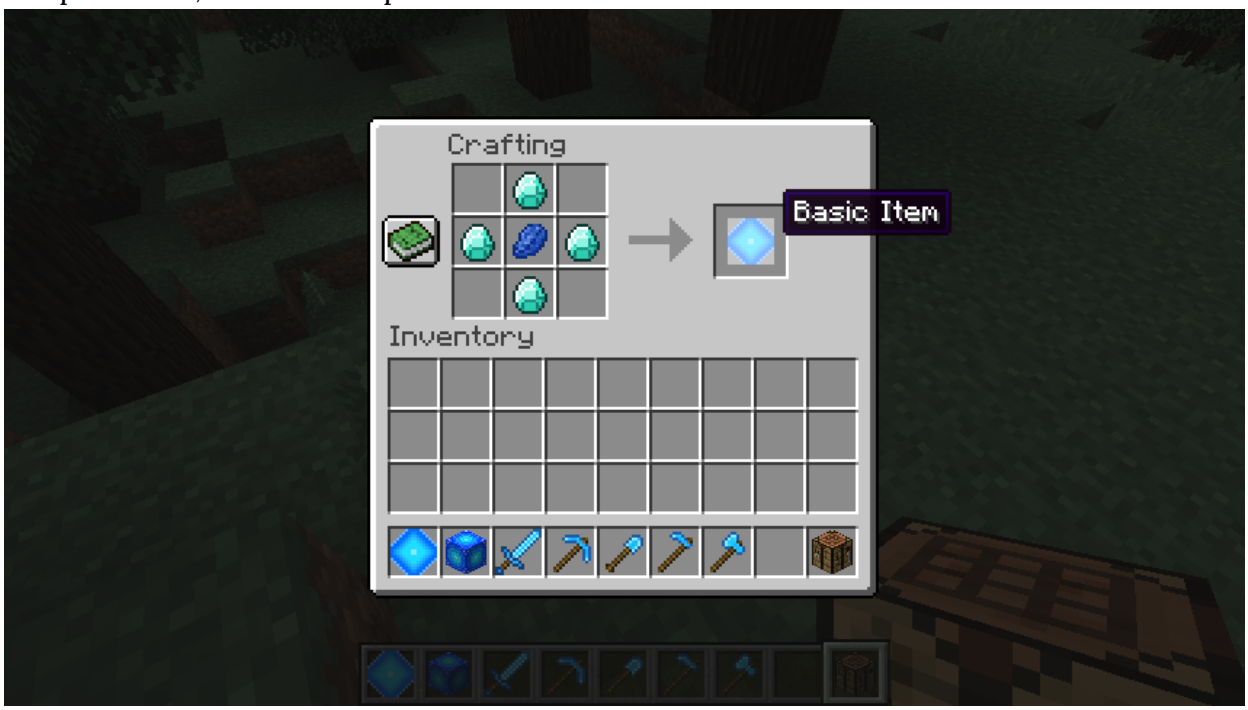
```
{
  "type": "minecraft:crafting_shapeless",
  "ingredients": [
    {
      "item": "minecraft:blaze_powder"
    },
    {
      "item": "minecraft:slime_ball"
    }
  ],
  "result": {
    "item": "minecraft:magma_cream"
  }
}
```

The file defines the recipe type as shapeless, lists the ingredients, and shows the result.

Now that we've gone over crafting recipes, let's add our own. In the `assets/tutorialmod/recipes` folder, create a new file named `first_item.json`. The name of the recipe file should be the same as the registry name of the result.

```
{
        "type": "minecraft:crafting_shaped",
        "pattern": [
                " X ",
                "X#X",
                " X "
        ],
        "key": {
                "X": {
                        "item": "minecraft:diamond"
                },
                "#": {
                        "item": "minecraft:dye",
                        "data": 4
                }
        },
        "result": {
                "item": "tutorialmod:first_item"
        }
}
```

Note that you need to fully-qualify the `type` of the recipe, even though vanilla doesn't (`crafting_shaped -> minecraft:crafting_shaped`) That's all for crafting recipes! Next, furnace recipes!



 (/feed.xml)  (https://github.com/cubicoder)

cubicoder • 2019 • Home (https://cubicoder.github.io)

Theme by beautiful-jekyll (https://deanattali.com/beautiful-jekyll/)