Minetest Mods

Skills/Concepts

This module provides an introduction to Minetest modding. It will cover some basic image manipulation and rudimentary Lua scripting.

The following concepts will be touched upon in this module:

- console use
- image editing
- basic usage of Minetest
- modding

Console Use

This module relies heavily on the command-line. You will be using a terminal emulator to execute the following commands.

Command	Description
cp	Copy file command
gimp	Can be used to launch the GNU Image Manipulation Program from a terminal instead of using the menus.

Overview

Modules for Minetest require a specific directory hierarchy. The files consist of an entry-point script for initialization and supporting files. Generally the supporting files consist of additional scripts, PNG files, documentation, etc. Although Minetest is written in C++, the scripting language uses Lua. Minetest provides hooks for sophisticated eventing and other application programmer interfaces (APIs).

We will be re-creating the directory hierarchy and creating a simple node that can be added to the world. Then we'll create three more nodes, the three additional nodes will make use of Minetest's eventing APIs to replace themselves with each previous node, down to the original node. Then the original node will replace itself with fire.

We will create the graphics with the Gimp (GNU Image Manipulation Program) and test each new addition in a standalone game where we can use Creative (or Administrative) powers to add the nodes to the world.

Process

Create the Graphics

0. Launch the Gimp, either from a menu through the graphical interface, or by typing `gimp` into a terminal.

1. Create a new image, dimensions 32x32. You'll need to click the turn-key arrow to choose a background color.
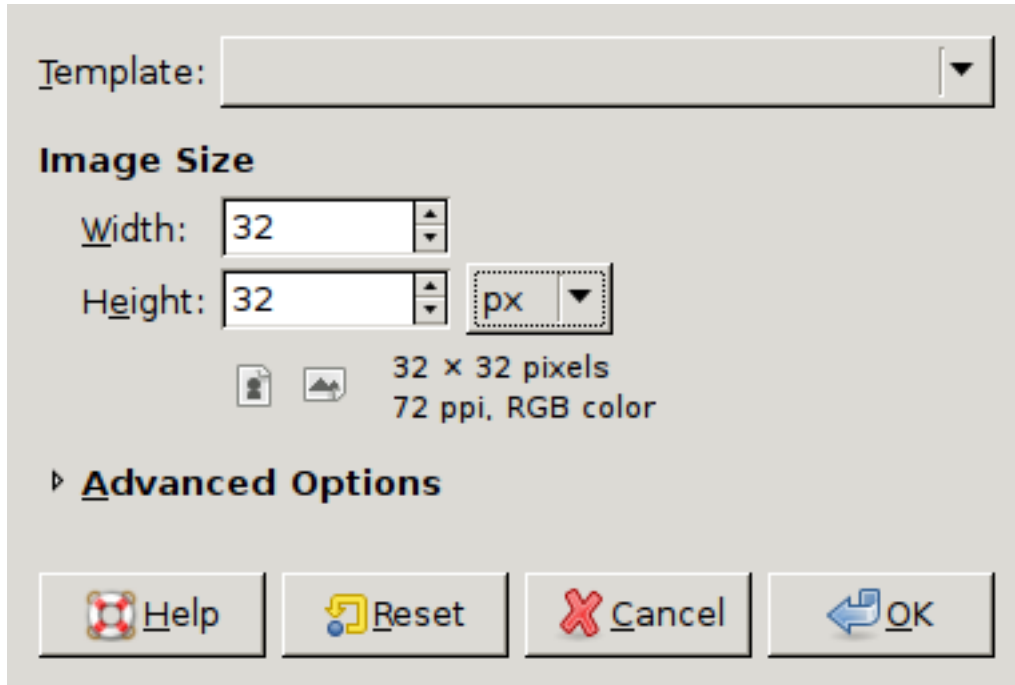

Figure 1: New Image Dialog

2. Choose either a background color or set it to transparent.

3. Hit *Control+B* to open the toolbox.

4. Chose the pencil tool.

5. Modify the properties of the tool to be square in dimensions.

6. Modify the size to be one pixel.

7. Zoom in on the image and color it in some distinguishable way. When you are done modifying it, save it, and then export it as a PNG. Save it where you like, but remember the location, for example, on the Desktop. Choose any name you like, this module will assume the name `final.png`.

8. Create three more graphics, containing the numbers 3, 2, and 1. Be sure to export these as PNGs.

Create the Base Module

The commands in this section should be executed from a terminal in the `minetest/mods` directory.

0. Create a directory to hold the module.

   mkdir mymodule

Figure 2: New Image Dialog, showing color options
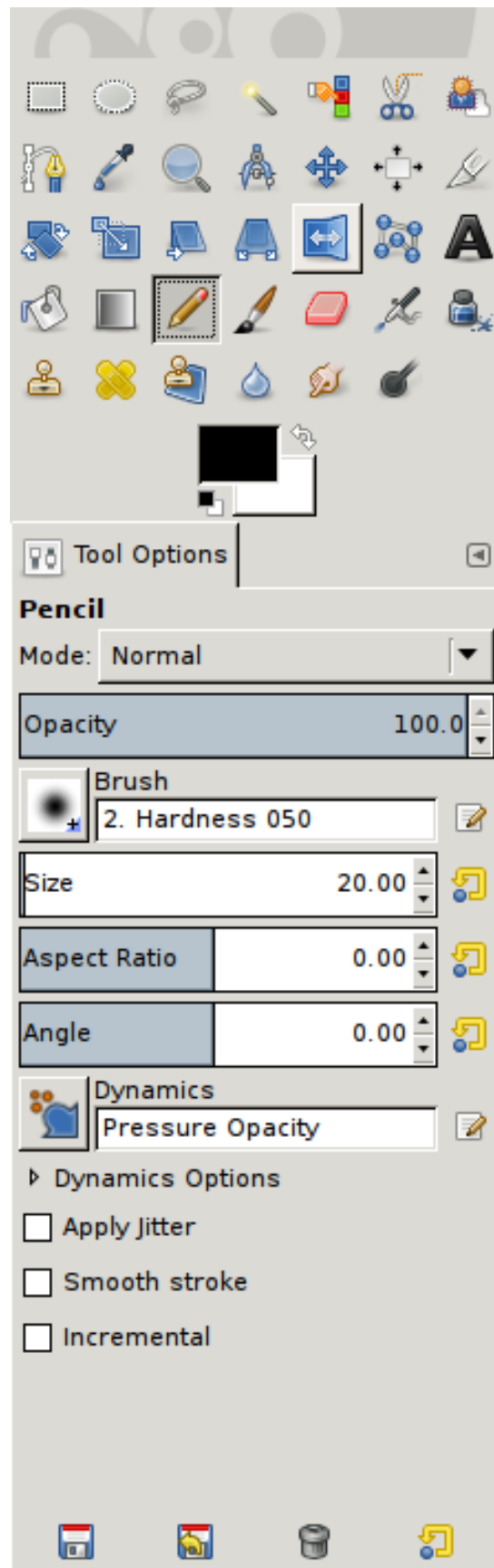
Figure 3: Toolbox
4
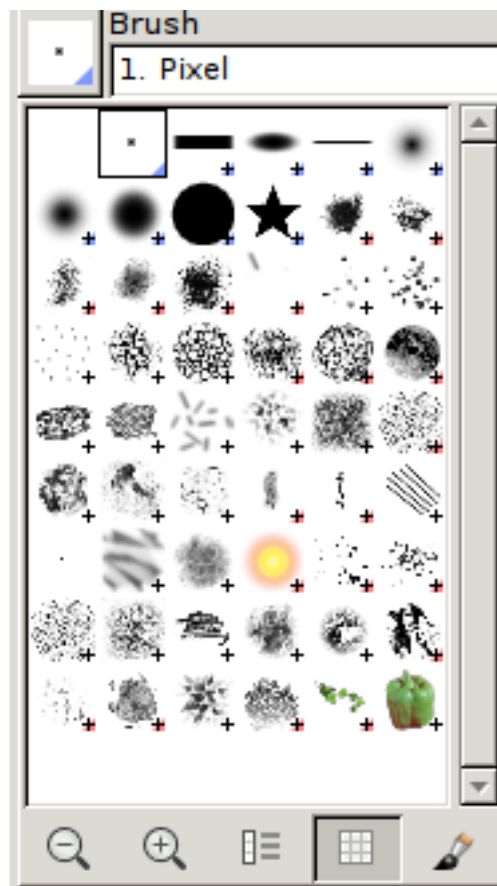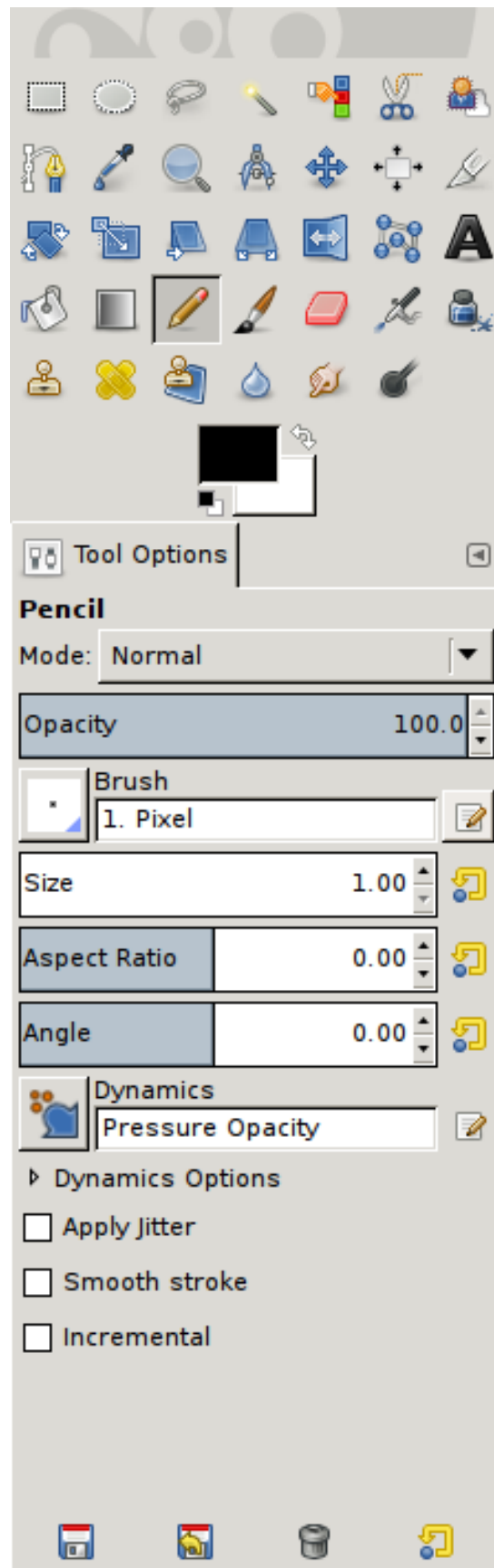
Figure 4: Brush Style

Figure 5: Pencil Size

1. Create a directory for the textures.

   mkdir mymodule/textures

2. Copy the image created earlier into the module's textures directory.

   cp ~/Desktop/final.png mymodule/textures

3. Create the entry-pont script, `mymodule/init.lua`, and put the following text in it:

```
minetest.register_node("mymodule:phos_final",
    {
        description = "phosphorous",
        tiles = {"final.png"},
        groups = {snappy=1, crumbly=2, },
        is_ground_content = false,
    }
)
```

4. Run the game, create a new Local Game.

5. Configure the new local game to enable the custom module.

6. Enter the game (in creative mode).

7. Press I.

8. Click the Node tab.

9. Navigate the node tab (it was page six on my system) and find the node with the graphic you created.

10. Drag it to one of the numbered slots, and place it in the world.

Make it Burn

0. Edit the `mymodule/init.lua` file and put the following in it:

```
minetest.register_node("mymodule:phos_final",
    {
        description = "phosphorous",
        tiles = {"final.png"},
        groups = {snappy=1, crumbly=2, },
        is_ground_content = false,
    }
)

minetest.register_abm({
    nodenames = {"mymodule:phos_final"},
    interval = 10,
    chance = 1,
    action = function(pos)
        minetest.add_node(pos, {name="fire:basic_flame"})
    end,
})
```

1. Press escape and choose *Exit to Menu* to leave the game and return to the menu.

2. The module is already active, so simply join the game again and add more of he custom block.

Make it Count

0. Edit the `mymodule/init.lua` file again. Copy and paste the entry for the node three times, rename the nodes to `mymodule:phos_1` to `mymodule:phos_3`.

1. Edit the tiles to refer to images `1.png` to `3.png`.

2. Copy the PNG files to the textures directory:

   ```
   cp ~/Desktop/1.png mymodule/textures/
   cp ~/Desktop/2.png mymodule/textures/
   cp ~/Desktop/3.png mymodule/textures/
   ```

3. Copy the `register_abs` block three times, modify the nodenames for each to refer to the node names from step 0.

4. Modify the `fire:basic_flame` entry for each of the new blocks and instead have them change into `mymodule:phos_2`, `mymodule:phos_1`, and `mymodule:phos_final`. Be sure to edit the description so it's easier to find the blocks in creative mode.

5. Reload the module, and this time add block three to the world and see it count down.

Troubleshooting and Errors

Place `CONTENT_IGNORE`

If you mistype the name of the module or the node, you may see an error that looks like the following:

```
2017-10-16 20:52:12: ERROR[Server]: Map::setNode(): Not allowing to place CONTENT_IGNORE while tryi
```

Module name must match

The prefixes used in the node name must be the same as the directory in which the module resides. You may see an error like the following: ~ 2017-10-16 22:27:17: ERROR[Main]: ModError: Failed to load and run script from /home/stephen/minetest/bin/../mods/mine/init.lua: 2017-10-16 22:27:17: ERROR[Main]: . . . me/stephen/minetest/bin/../builtin/game/register.lua:62: Name wrong:phos_final does not follow naming conventions: "mine:" or ":" prefix required 2017-10-16 22:27:17: ERROR[Main]: stack traceback: 2017-10-16 22:27:17: ERROR[Main]: [C]: in function 'error' 2017-10-16 22:27:17: ERROR[Main]: . . . me/stephen/minetest/bin/../builtin/game/register.lua:62: in function 'check_modname_prefix' 2017-10-16 22:27:17: ERROR[Main]: . . . me/stephen/minetest/bin/../builtin/game/register.lua:110: in function 'register_item' 2017-10-16 22:27:17: ERROR[Main]: . . . me/stephen/minetest/bin/../builtin/game/register.lua:177: in function 'register_node' 2017-10-16 22:27:17: ERROR[Main]: /home/stephen/minetest/bin/../mods/mine/init.lua:2: in main chunk 2017-10-16 22:27:17: ERROR[Main]: Check debug.txt for details. ~

Make sure the node name prefix matches the directory name. In the stack trace above, the `wrong` should be `mine`. The timestamp that reads `2017-10-16 22:27:17` in the above output contains the correct directory name and timestamp `2017-10-16 22:27:17` contains both the incorrect and the correct name.

Conclusion

We created very simple graphics and placed them in the correct directory hierarchy and made a simple Lua script that drove everything. Then we loaded up the module in minetest. We were able to see that reloading the code to try new things only required leaving the world and re-entering.