NetworkCraft

Members
Sam Spalding
Chris Van Essen

Sponsor
UI CoE: Daniel Conte de Leon

Objective

For this project, we were asked to develop a Minecraft Mod that enables realistic modeling of TCP/IP networks for research and instruction. Our goal was to create custom blocks that would act as basic devices you'd find in a network, such as a desktop, a switch, and network wires. With these custom blocks, we set out to implement a simulation of physical Layer 1 properties. i.e. Allow Desktop blocks to send data to one another when connected to a Switch block.

Installing Software

MCreator

MCreator is the main software used to create our Minecraft Mod. It's an open-source software used to make Minecraft Java Edition mods, Bedrock Edition Add-Ons, and data packs using an intuitive easy-to-learn interface or with an integrated code editor. To download MCreator, go to this link: https://mcreator.net/ and click on the "Download MCreator" button towards the top of the screen. Our team worked in MCreator v.2020.5, so if the next team plans to continue where we left off, use this version. However, if the next team wishes to start over, then it would be best to select the most recent version of MCreator. You'll be given different file types to download depending on your OS. Once the file is downloaded, simply follow the install wizard to your preference. Once the software is installed, select the "New Workspace" option to get started on your mod.

Setting Up a GitHub Repository and Installing GitHub Desktop

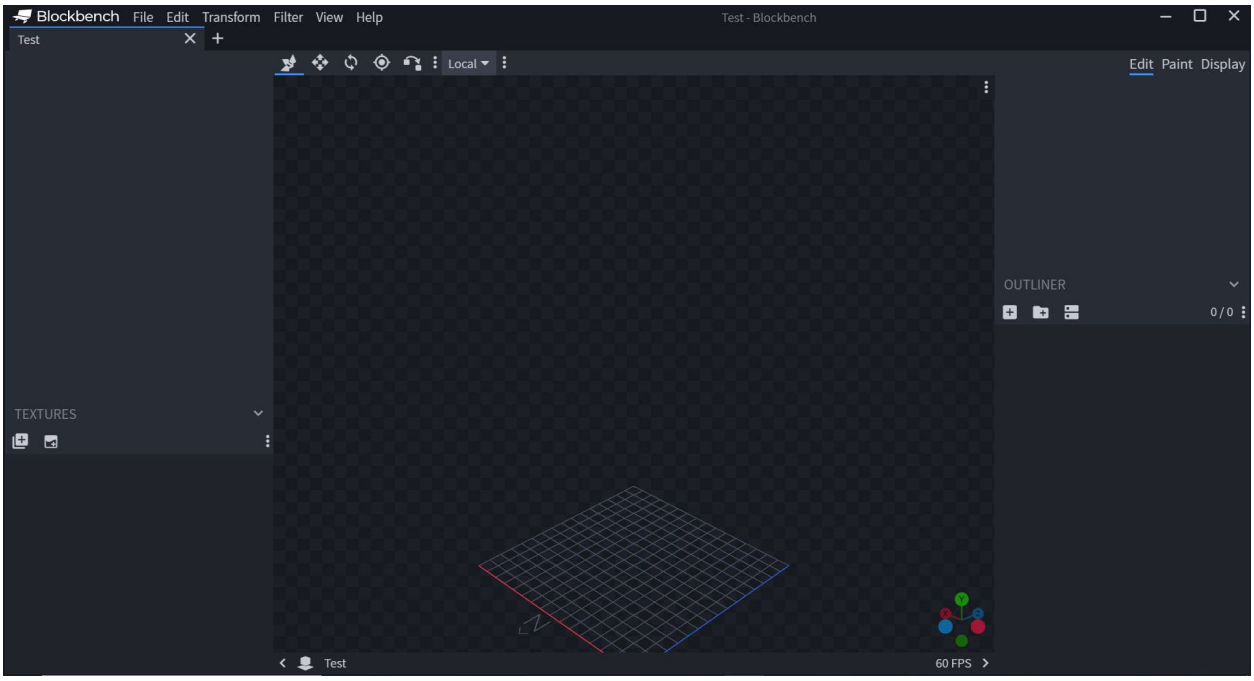
To make sure all team members can see others work and collaborate efficiently, using GitHub is the best choice. MCreator has a repository upload system built into its software, but it is unreliable, so it's recommended to install GitHub Desktop to do your file syncs. Go to this link to install GitHub Desktop: https://desktop.github.com/ and select the download for your OS. Once GitHub Desktop is installed go to File then add local repository. Search for the path that connects to your MCreator workspace and select it. Now you have your MCreator workspace hooked up to a GitHub repository. To share the repository to other team members, you'll need to do so on the GitHub website. You can either type in the URL manually, or you can select "Repository" then "View on GitHub" on the desktop version. From there, go to Settings and Manage Access and simply add people to collaborate with.

BlockBench

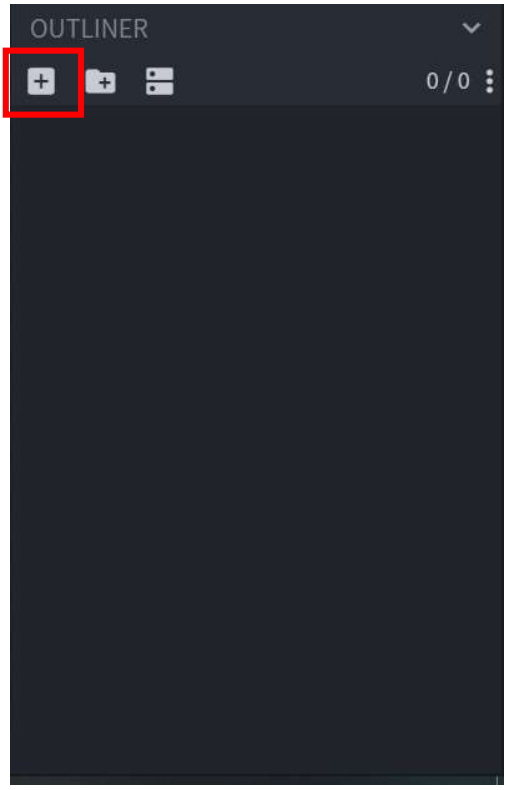
BlockBench is a free, modern model editor for low-poly and boxy models with pixel art textures. This software was used to create our custom block models. To install BlockBench, go to this link: https://www.blockbench.net/downloads and choose the download file for your OS. Follow the wizard and select your preferences and once that is completed you can start creating your custom Minecraft blocks. Creating blocks and exporting them will be covered later in this document.

Creating a Custom Block in BlockBench

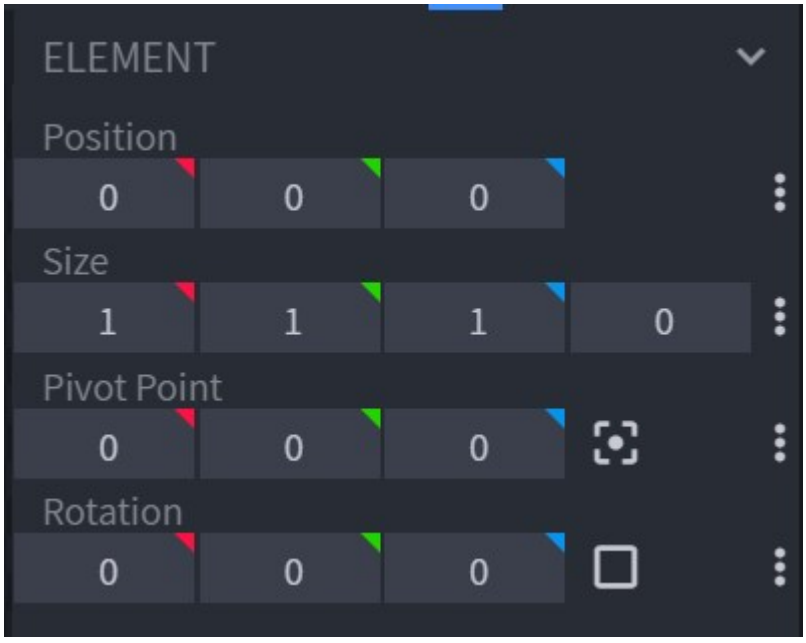
Open BlockBench and select Java Block/Item so create a block for Minecraft. Name the block and select Confirm. You'll be a given a editor that looks like this:



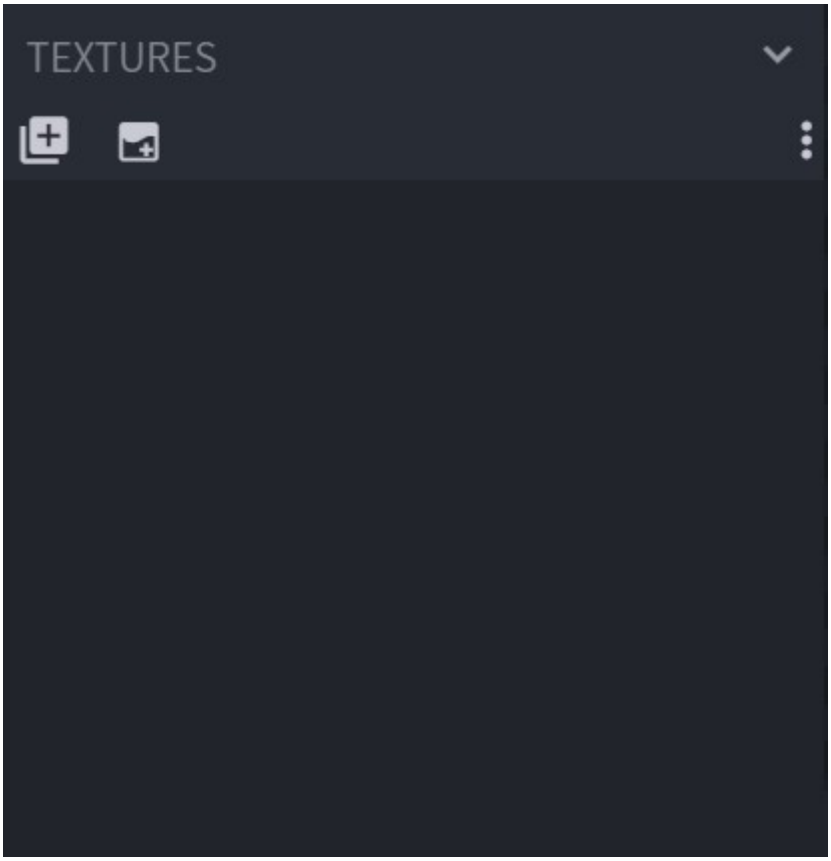
From here go to the right pane and select the square icon with the plus sign in it to add a cube to the editor.



Once a cube is placed, a new pane is opened on the right side of the editor that allows you to manipulate the cube properties.

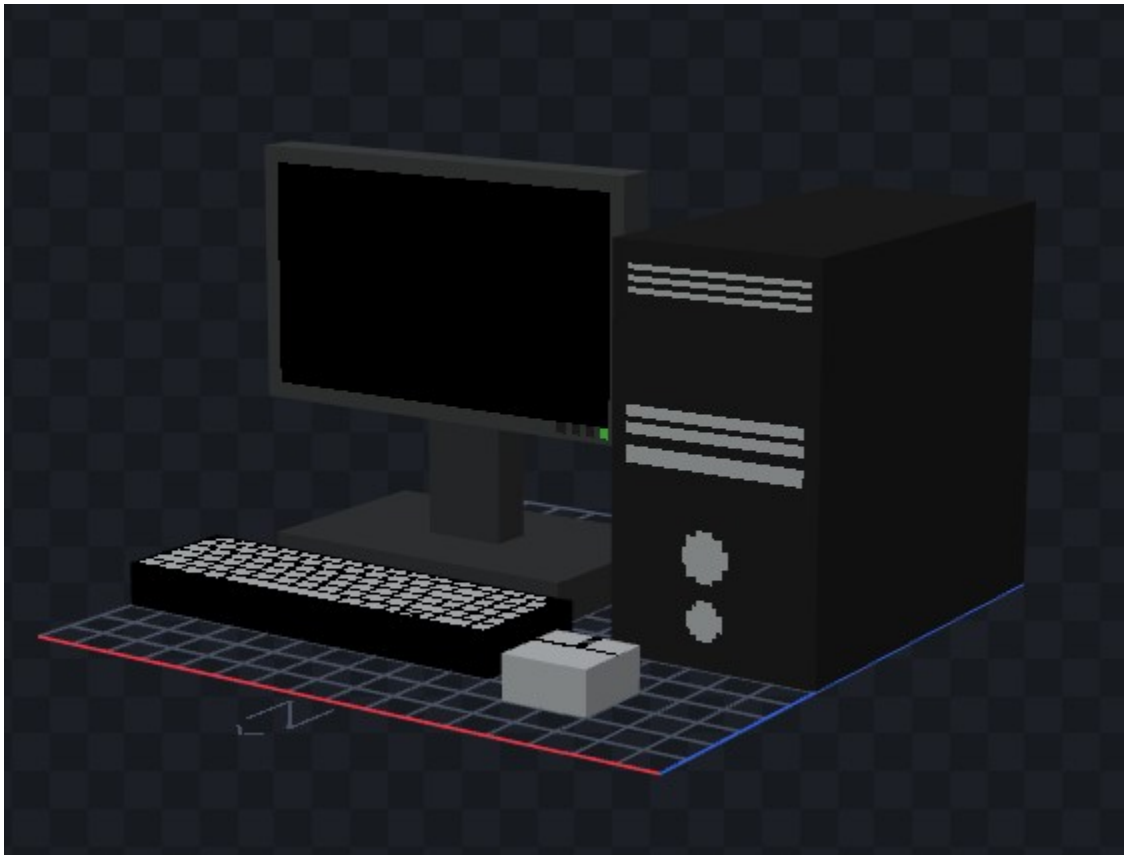


You can add several cubes to the editor to create complex models. Another feature of BlockBench is that you can apply textures to your block faces. The pane on the bottom left is where you can import and create your own textures. The left icon is to import textures into BlockBench while the right icon opens BlockBench's texture editor.



BlockBench's texture editor is lacking, so its recommended to use a different resource for texture creation. A good website that was used for our texture creation was https://www.pixilart.com/draw. It's a more robust editor than BlockBench and it allows you to save your created textures, which can then be imported into BlockBench and easily applied to a block face. If you use pixilart, just know that once you are done with a texture go to File>Export/Download and select Download .PNG since BlockBench only accepts .png files for import.

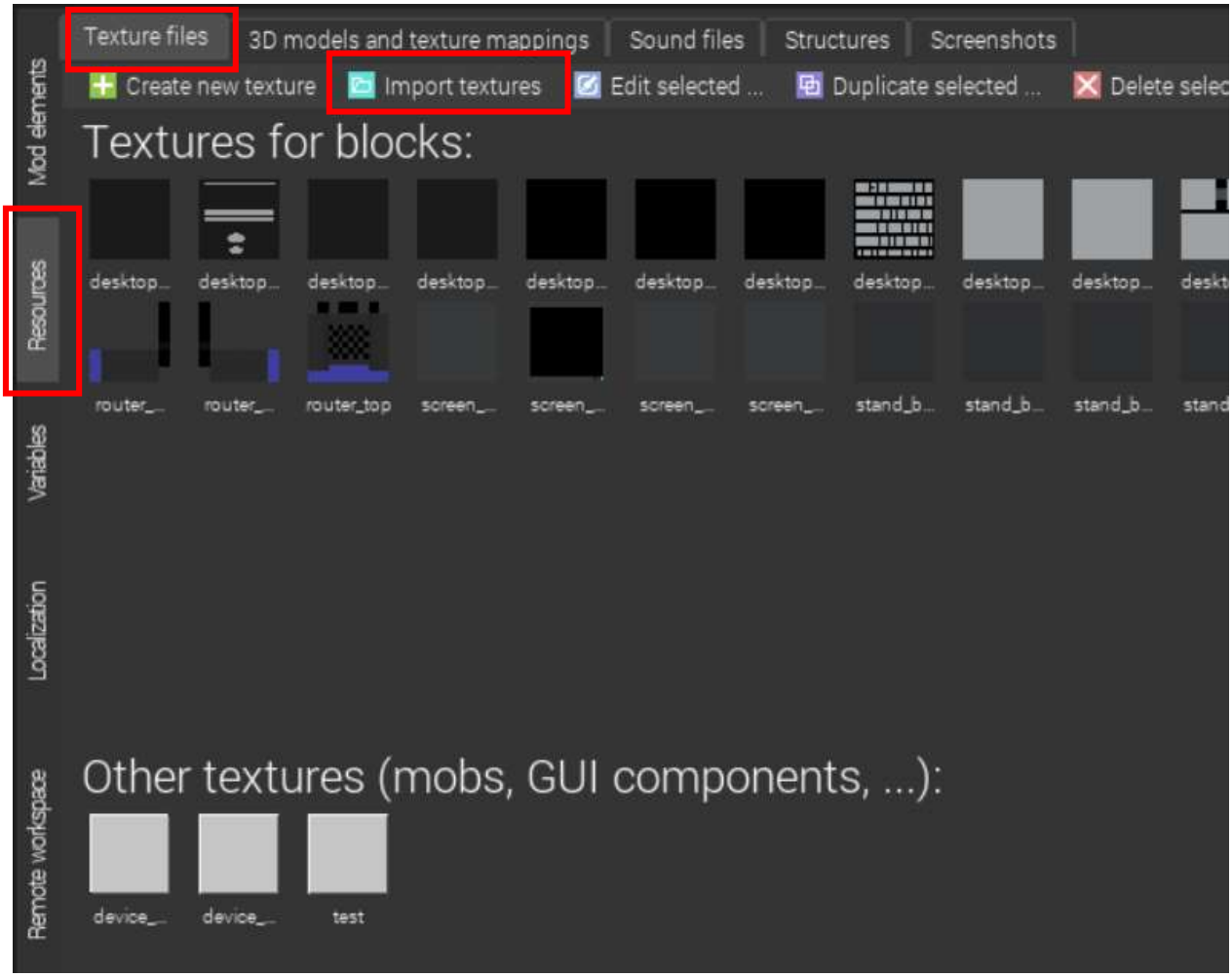
The best example of our use in Block Bench is our Desktop Block.



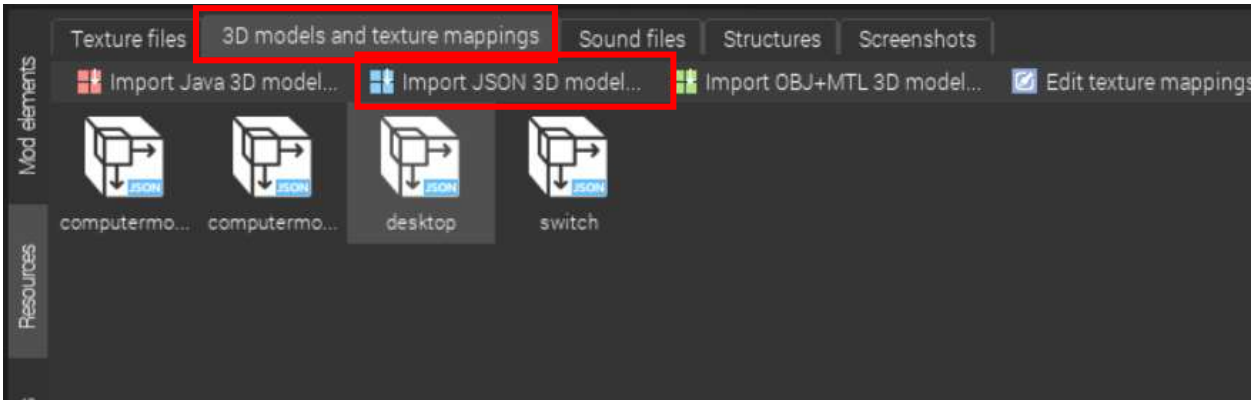
This block was created with a total of 6 separate cubes and all textures were created with pixilart.

Exporting Custom Block from BlockBench to MCreator

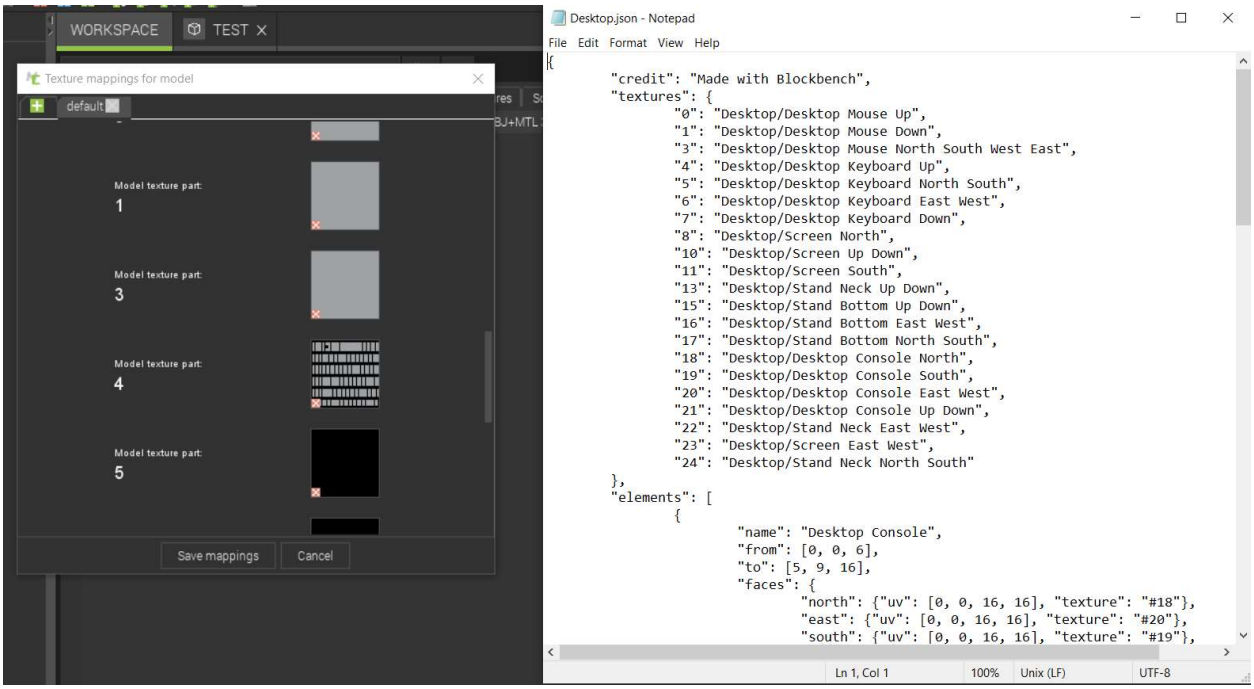
When you have a block completely done in BlockBench you'll need to go through a series of steps to get it into MCreator without any issues. First thing you'll need to do is save your model as a .bbmodel file just so you'll be able to edit your block anytime. Then you'll go to File>Export>Export Block/Item Model, which will save to your device as a .json file. Once that's done, open MCreator and go to Resources on the left side bar and then to Texture files.



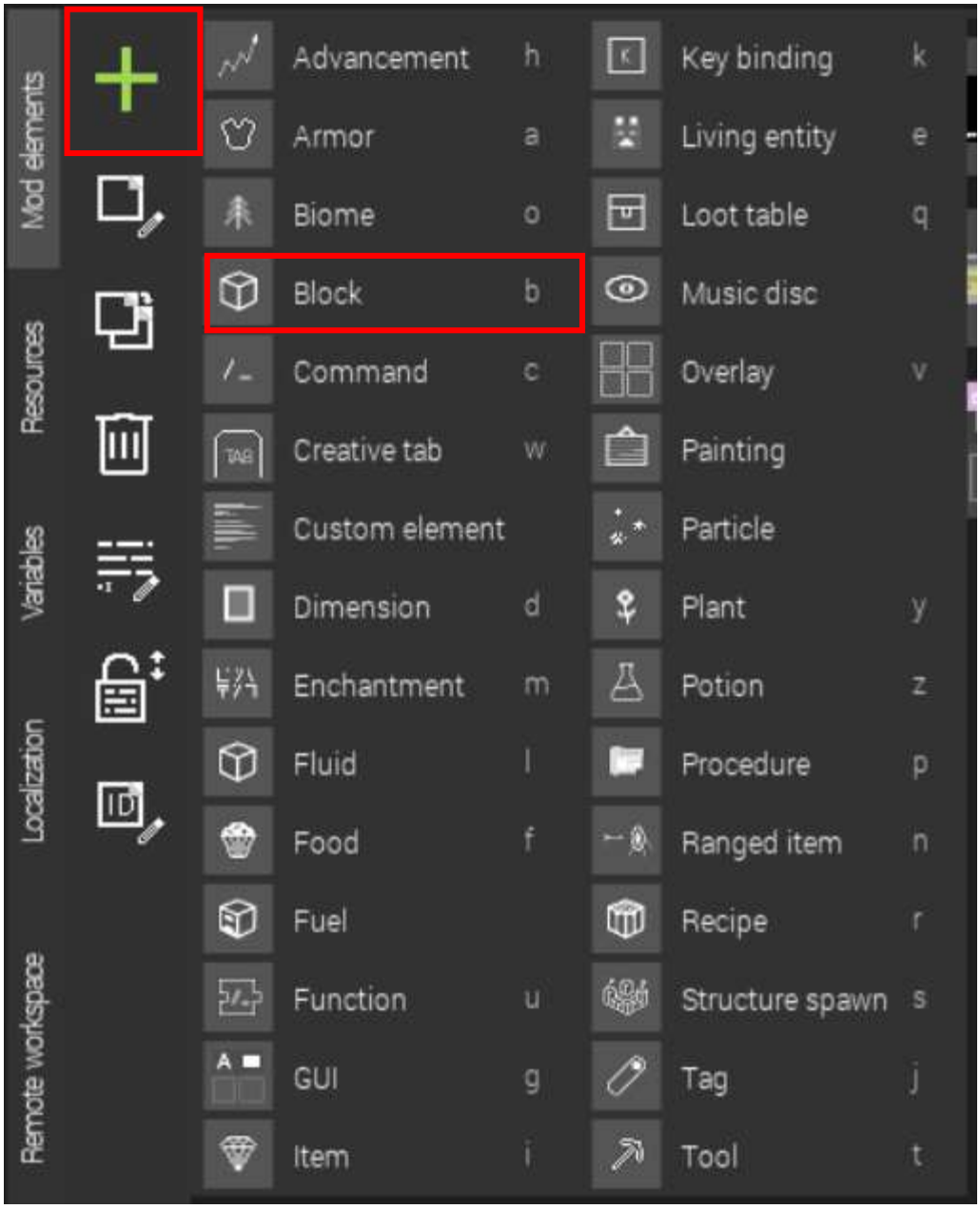
Upload all those textures that was made in pixilart and name them accordingly. Then go to 3D models and texture mapping and select Import JSON 3D model. There you can upload that .json file we saved.



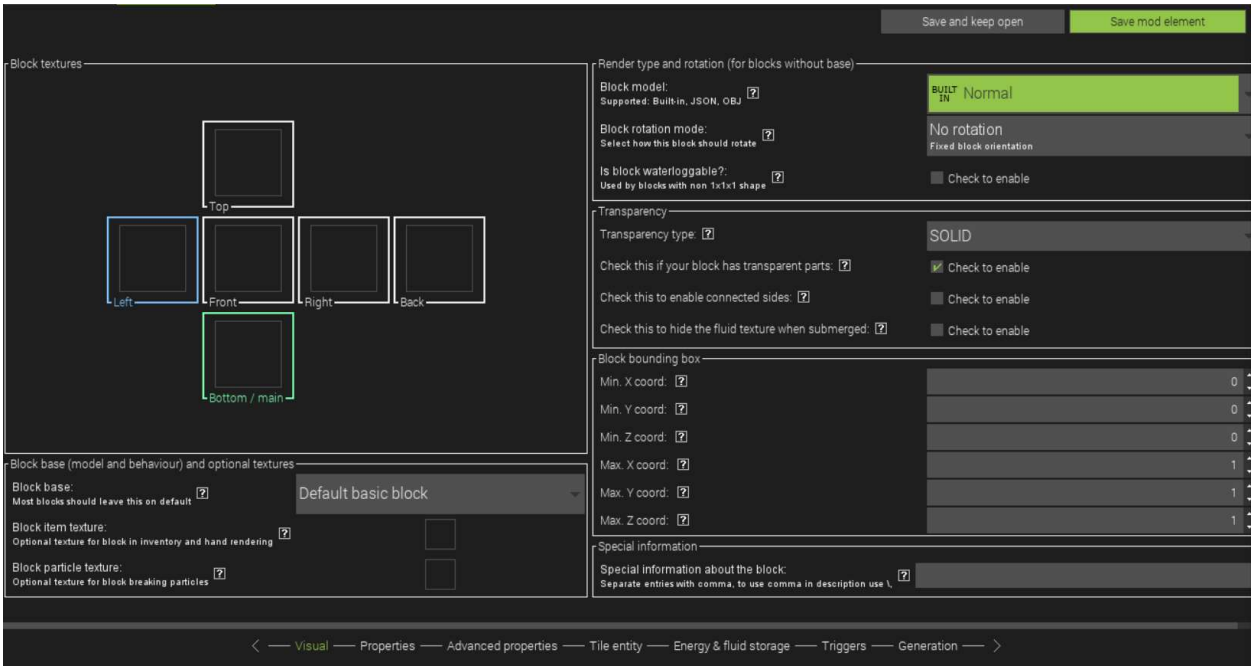
Next you'll double click the newly imported json file where you can edit the texture mappings. Here is where it can be a little messy. Go to your file directory and open that json file into a word editor like Notepad or similar. Here you'll see that the textures you applied to your block are associated with numbers. The same is seen in MCreator. You'll need to map the textures you uploaded to MCreator to the corresponding numbers. The .json file will be your guide to tell you which texture goes with which number. This is why naming your textures is important.



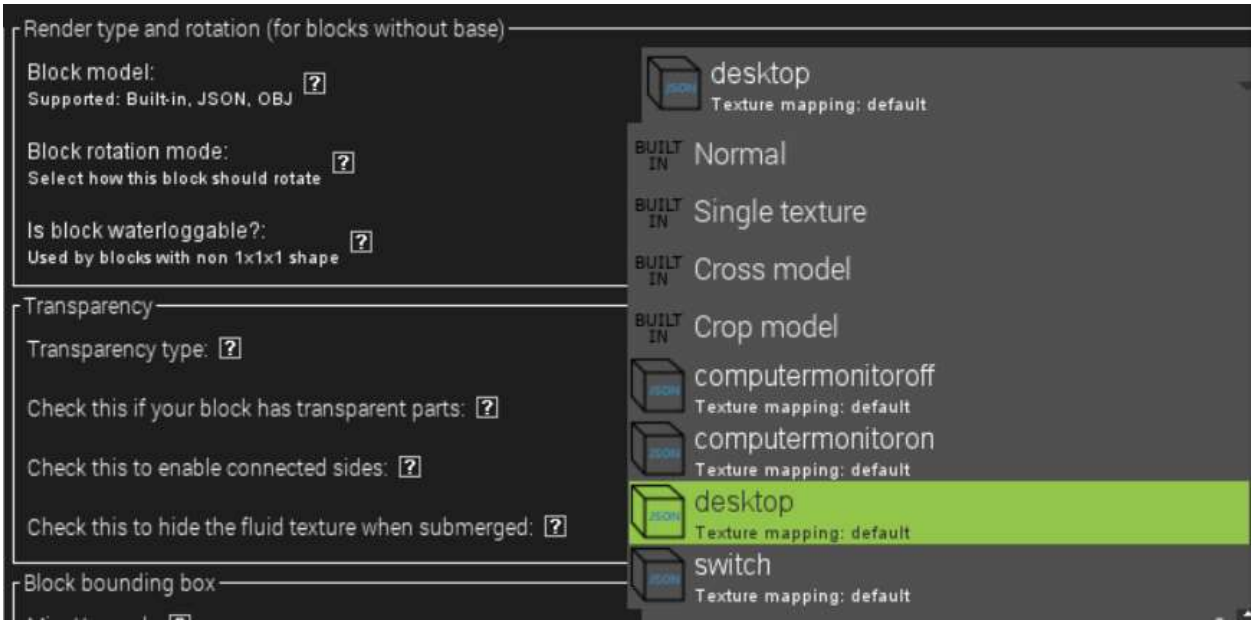
With that done, go back to Mod elements and select the plus icon to create a new Block element.



You'll be asked to name the block and then it will open the Block editor.



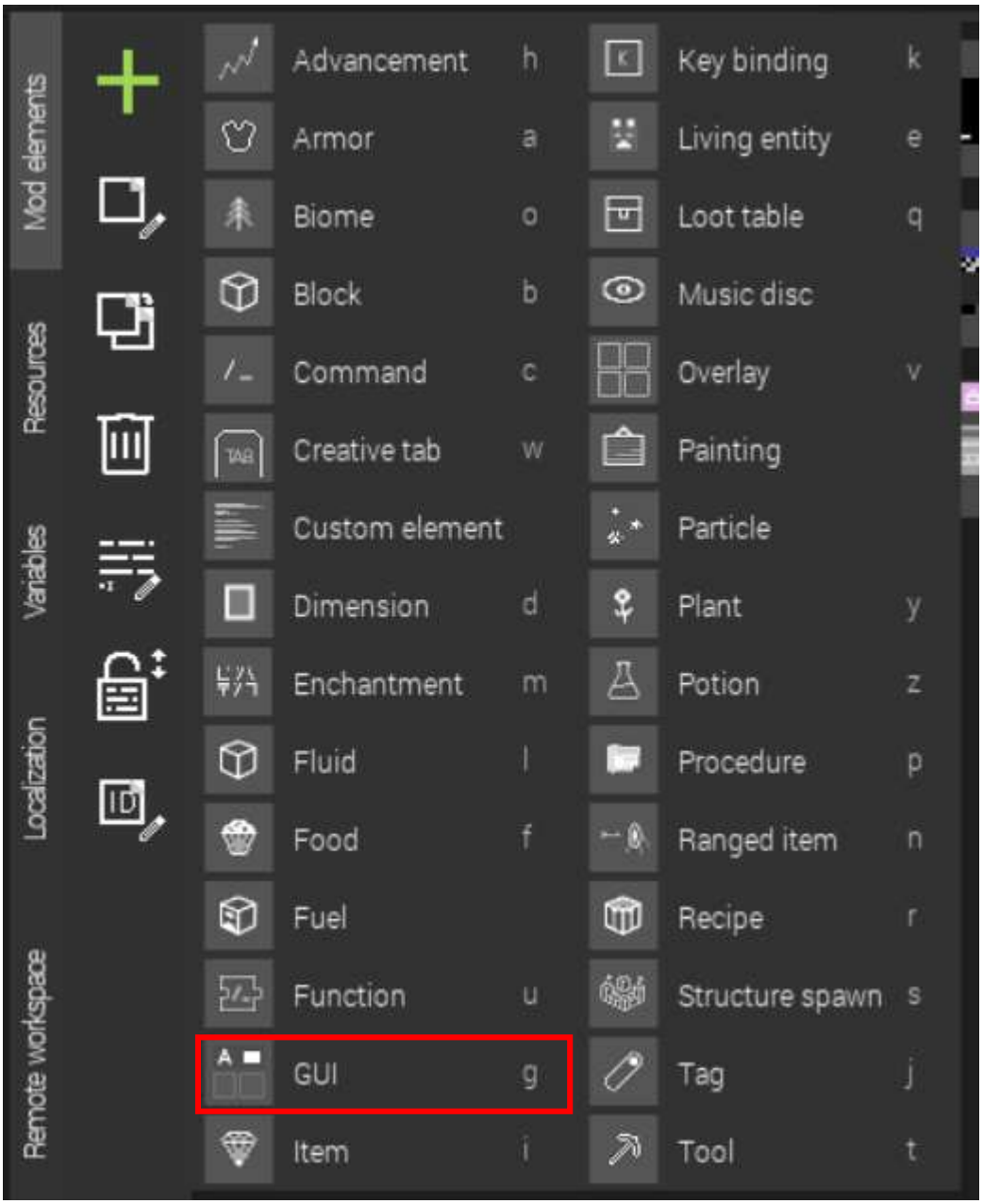
From here you'll want to select Block model under "Render type and rotation" in the upper right corner. Select the json file that was just imported and mapped.



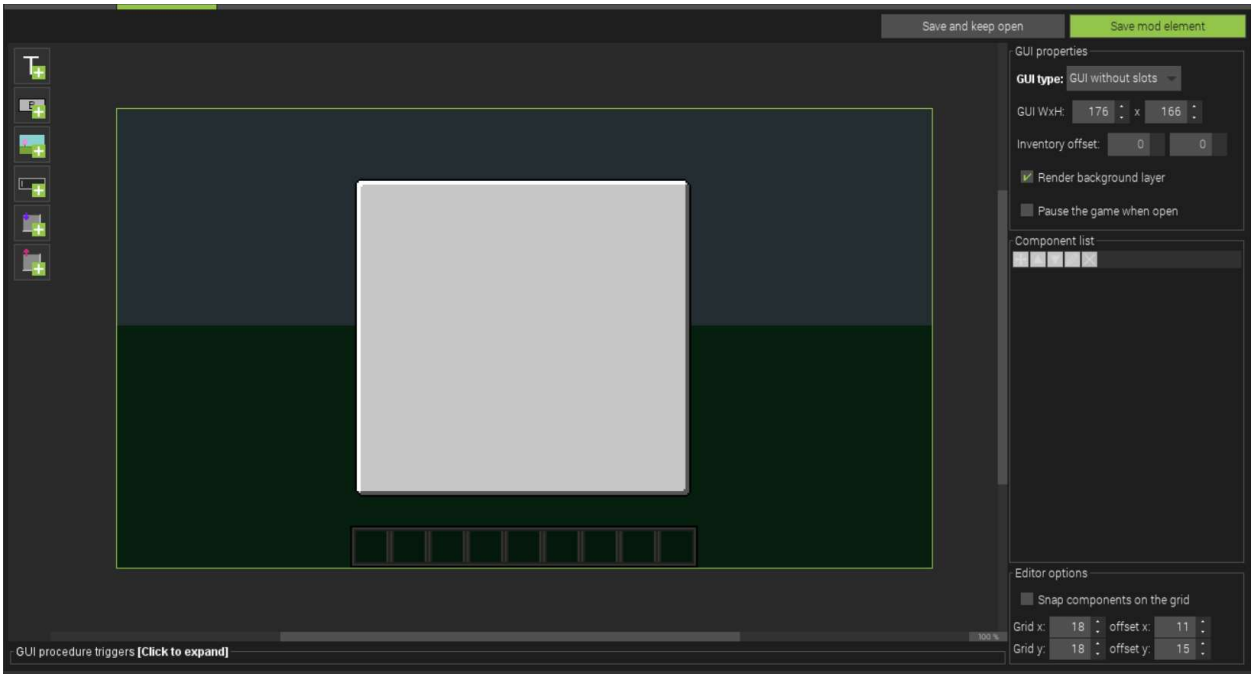
Doing this will make the block texture section to only allow you to place a texture on the Bottom/Main section. It doesn't matter which texture you put here because your texture mapping will handle it all so just choose a solid texture here. There are many other properties you can edit for the block, but for the Desktop block everything was left to default.

Creating a GUI in MCreator

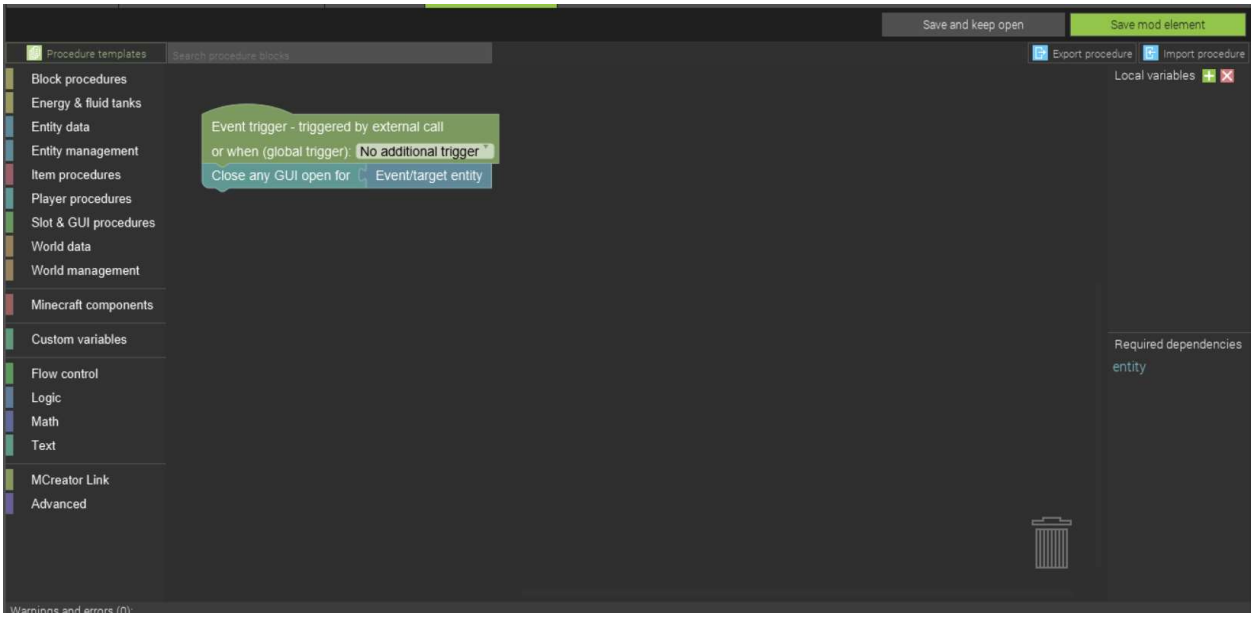
For our Desktop block, we created a GUI that would display Device Name, MAC address, and IP address for each placed block. First, we opened a GUI element.



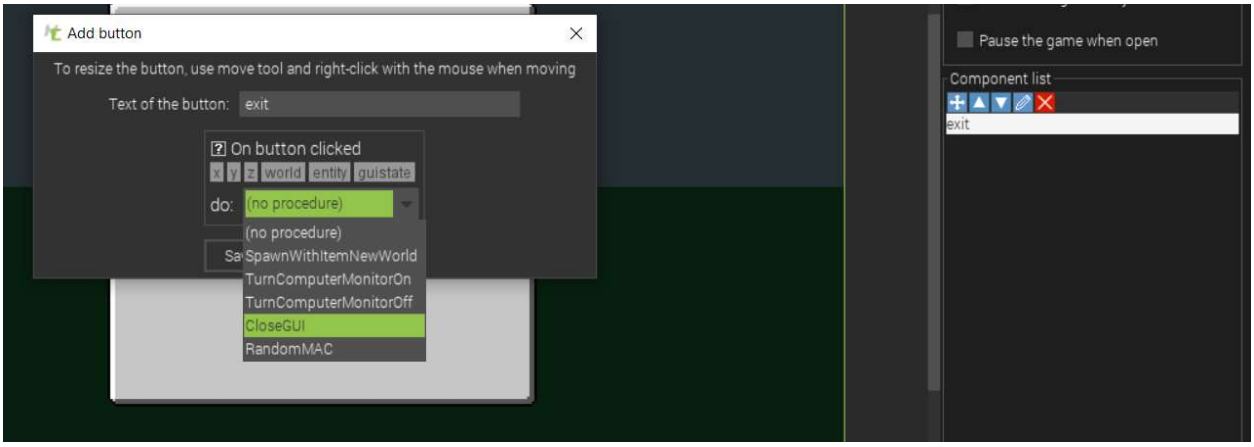
Once selected, you'll find an editor that looks like the following



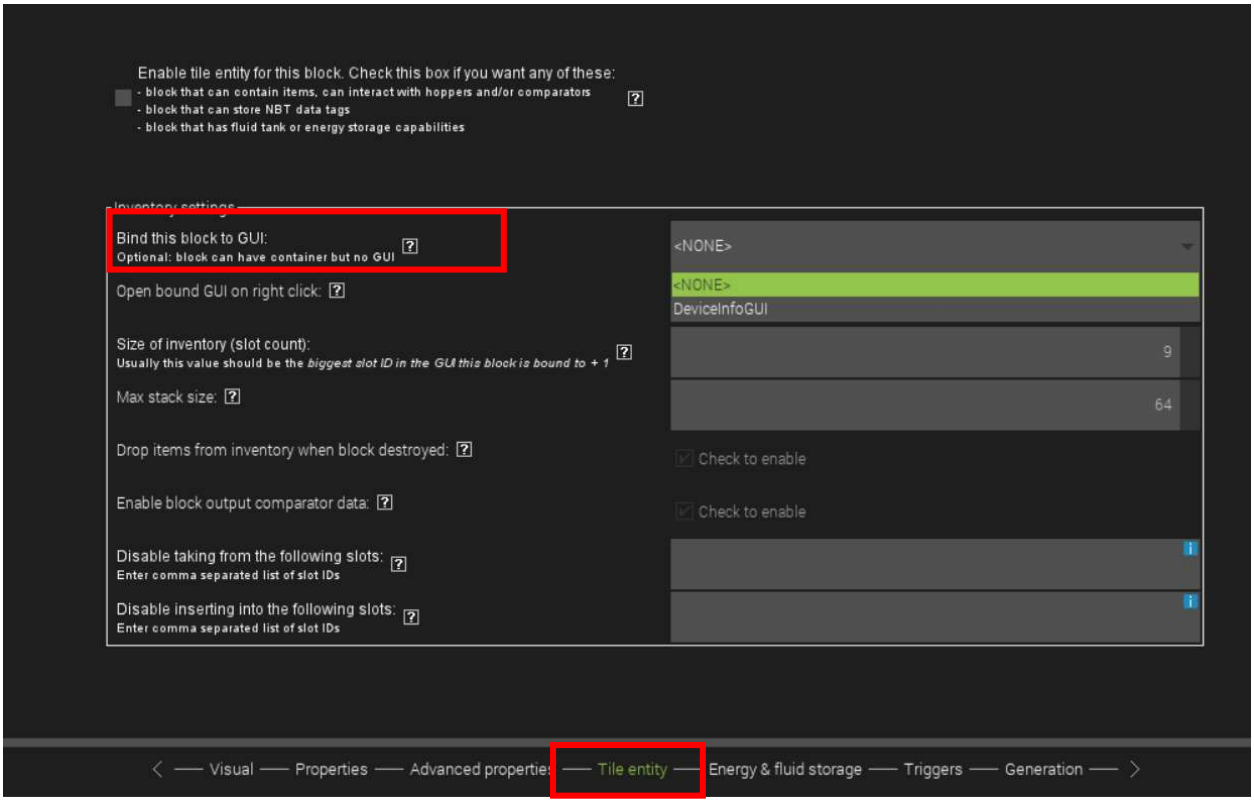
Here you can add text, buttons and images as you please. For example, to add an exit button is simple, but getting it to actually do something requires a procedure to be attached to the button. To do so, we created a simple Procedure element.



To attach this procedure to a button in the GUI, double click on the entity in the under the Component list on the right and it'll open a window. There you'll see you can choose a procedure that executes "On button clicked".

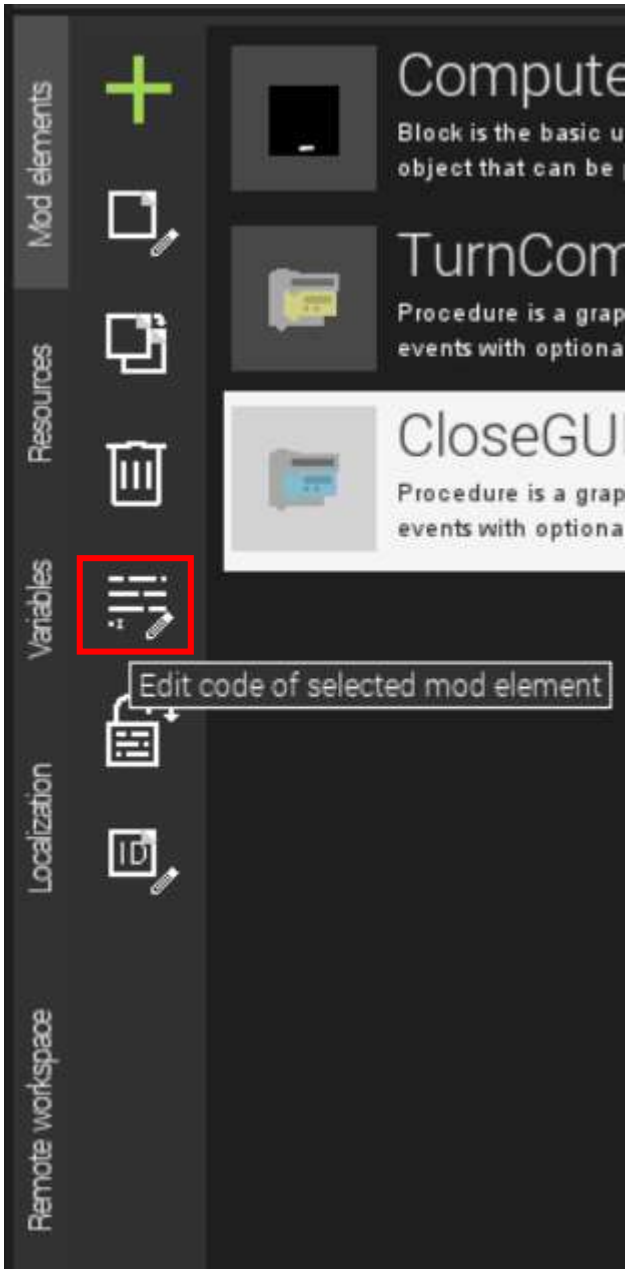


After getting the initial layout of the GUI figured out, we then applied the GUI to our Desktop block. Applying the GUI is done by opening the block editor again and going to Tile Entity where you can select the GUI that was just created. Also, check the box below it to make the GUI appear after right clicking on the block in-game.



Now the hardest part here was to get it so that each time a Desktop block was placed in the world a randomly generated MAC address and IP address would be assigned to the block and it would be displayed in the GUI. First, a Procedure element was created, but we didn't use MCreator's editor. You can open up the

code of a specific mod element by clicking on the icon on the left sidebar as shown:



This will open up MCreator's code editor and this is where we inserted a simple piece of code to generate random MAC addresses and save them in a string.

```
public static String getRandomMacAddress() {
    String mac = "";
    Random r = new Random();
    for (int i = 0; i < 6; i++) {
        int n = r.nextInt(255);
        mac += String.format("%02x", n);
    }
    System.out.println("Mac address generated: " + mac.toUpperCase());
    return mac.toUpperCase();
}
```

Note here that when you edit the code manually for an element, you must “lock” the element. Otherwise, MCreator will overwrite your work. Locking an element also prevents you from using the UI editor, so you are then stuck editing the raw code of the element. I’ll be honest from this point. I don’t remember exactly how we got it to start working. It was a lot of code combing and research. Below you’ll find the code for generating the random MAC and IP addresses and the edited code for the Desktop Block and GUI.

RandomMacAddress:

```
package net.mcreator.networkcraft.procedures;

import net.minecraft.world.IWorld;
import net.minecraft.util.math.BlockPos;
import net.minecraft.tileentity.TileEntity;
import net.minecraft.block.BlockState;

import
net.mcreator.networkcraft.NetworkcraftModElements;

import java.util.Map;
import java.util.Random;

@NetworkcraftModElements.ModElement.Tag
public class RandomMACProcedure extends
NetworkcraftModElements.ModElement {
    public RandomMACProcedure(NetworkcraftModElements
instance) {
        super(instance, 10);
    }
```

```
    public static void executeProcedure(Map<String,
Object> dependencies) {

    if (dependencies.get("x") == null) {
            if (!dependencies.containsKey("x"))
                System.err.println("Failed to load
dependency x for procedure RandomMACProcedure!");
            return;
        }
        if (dependencies.get("y") == null) {
            if (!dependencies.containsKey("y"))
                System.err.println("Failed to load
dependency y for procedure RandomMACProcedure!");
            return;
        }
        if (dependencies.get("z") == null) {
            if (!dependencies.containsKey("z"))
                System.err.println("Failed to load
dependency z for procedure RandomMACProcedure!");
            return;
        }
        if (dependencies.get("world") == null) {
            if (!dependencies.containsKey("world"))
                System.err.println("Failed to load
dependency world for procedure RandomMACProcedure!");
            return;
        }
        double x = dependencies.get("x") instanceof
Integer ? (int) dependencies.get("x") : (double)
dependencies.get("x");
        double y = dependencies.get("y") instanceof
Integer ? (int) dependencies.get("y") : (double)
dependencies.get("y");
        double z = dependencies.get("z") instanceof
Integer ? (int) dependencies.get("z") : (double)
dependencies.get("z");
```

```
        IWorld world = (IWorld)
dependencies.get("world");
        if (!world.getWorld().isRemote) {
            BlockPos _bp = new BlockPos((int) x, (int)
y, (int) z);
            TileEntity _tileEntity =
world.getTileEntity(_bp);
            // Copied above, but as another variable
for the IP address
            TileEntity _tileEntity2 =
world.getTileEntity(_bp);
            BlockState _bs = world.getBlockState(_bp);
            if (_tileEntity != null)

    _tileEntity.getTileData().putString("macAddress",
getRandomMacAddress());
                // Copied above, but edited for IP
address

    _tileEntity2.getTileData().putString("ipAddress",
getRandomIPAddress());
            world.getWorld().notifyBlockUpdate(_bp,
_bs, _bs, 3);
        }
    }


    public static String getRandomMacAddress() {
        String mac = "";
        Random r = new Random();
        for (int i = 0; i < 6; i++) {
            int n = r.nextInt(255);
            mac += String.format("%02x", n);
        }
        System.out.println("Mac address generated: " +
mac.toUpperCase());
        return mac.toUpperCase();
```

```
    }

    public static String getRandomIPAddress(){

    return "10." + (int) (Math.random() * 255) + "."
                + (int) (Math.random() * 255) + "."
                + (int) (Math.random() * 255);
    }
}
```

Desktop Block:

```
@Override
        public void onBlockAdded(BlockState state,
World world, BlockPos pos, BlockState oldState, boolean
moving) {
            super.onBlockAdded(state, world, pos,
oldState, moving);
            int x = pos.getX();
            int y = pos.getY();
            int z = pos.getZ();
            // Output for testing purposes
            System.out.println("Running onBlockAdded
procedure for desktop...");
            {
                Map<String, Object> $_dependencies =
new HashMap<>();
                // Added code
                $_dependencies.put("x", x);
                $_dependencies.put("y", y);
                $_dependencies.put("z", z);
                $_dependencies.put("world", world);
                // Added code end
RandomMACProcedure.executeProcedure($_dependencies);
            }
        }
```

Desktop GUI:

```
@Override
        protected void
drawGuiContainerForegroundLayer(int mouseX, int mouseY)
{
            this.font.drawString("Device Information",
67, 16, -12829636);
            this.font.drawString("Device Name", 12,
42, -12829636);
            this.font.drawString("MAC Address", 12,
73, -12829636);
            this.font.drawString("IP Address", 14,
102, -12829636);
            this.font.drawString("" + (new Object() {
                public String getValue(BlockPos pos,
String tag) {
                    TileEntity tileEntity =
world.getTileEntity(pos);
                    if (tileEntity != null)
                        return
tileEntity.getTileData().getString(tag);
                    return "";
                }
                // edited the in quotes below from
"tile entity" to "macAddress" which matches what
RandomMACProcedure names this as
            }.getValue(new BlockPos((int) x, (int) y,
(int) z), "macAddress")) + "", 80, 72, -12829636);
            // copied macAddress code above and edited
it for ipAddress
            this.font.drawString("" + (new Object() {
                public String getValue(BlockPos pos,
String tag) {
                    TileEntity tileEntity =
world.getTileEntity(pos);
                    if (tileEntity != null)
```

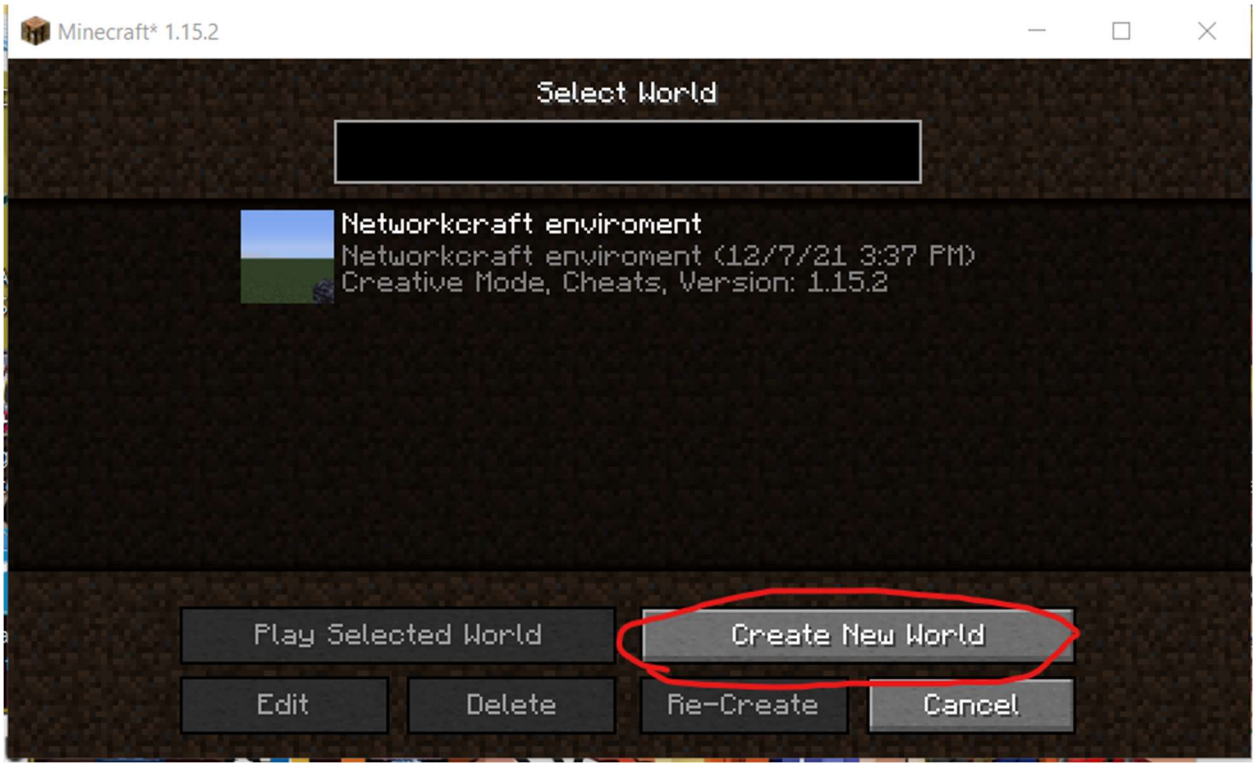
```
                                return
tileEntity.getTileData().getString(tag);
                            return "";
                        }
                    }.getValue(new BlockPos((int) x, (int) y,
(int) z), "ipAddress")) + "", 80, 99, -12829636);

            }
```

The end result of our GUI looks like the following:

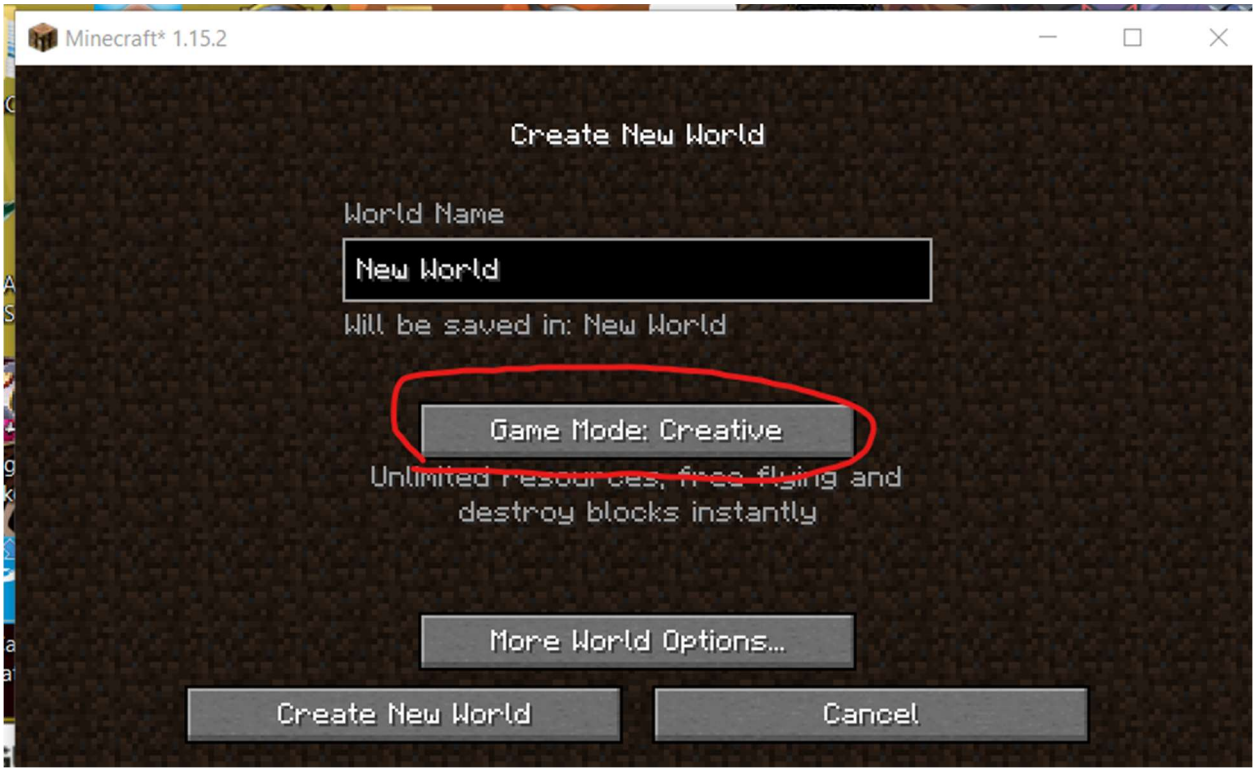
Minecraft Environment World Creation

So to make the environment for our project first I went to Single Player

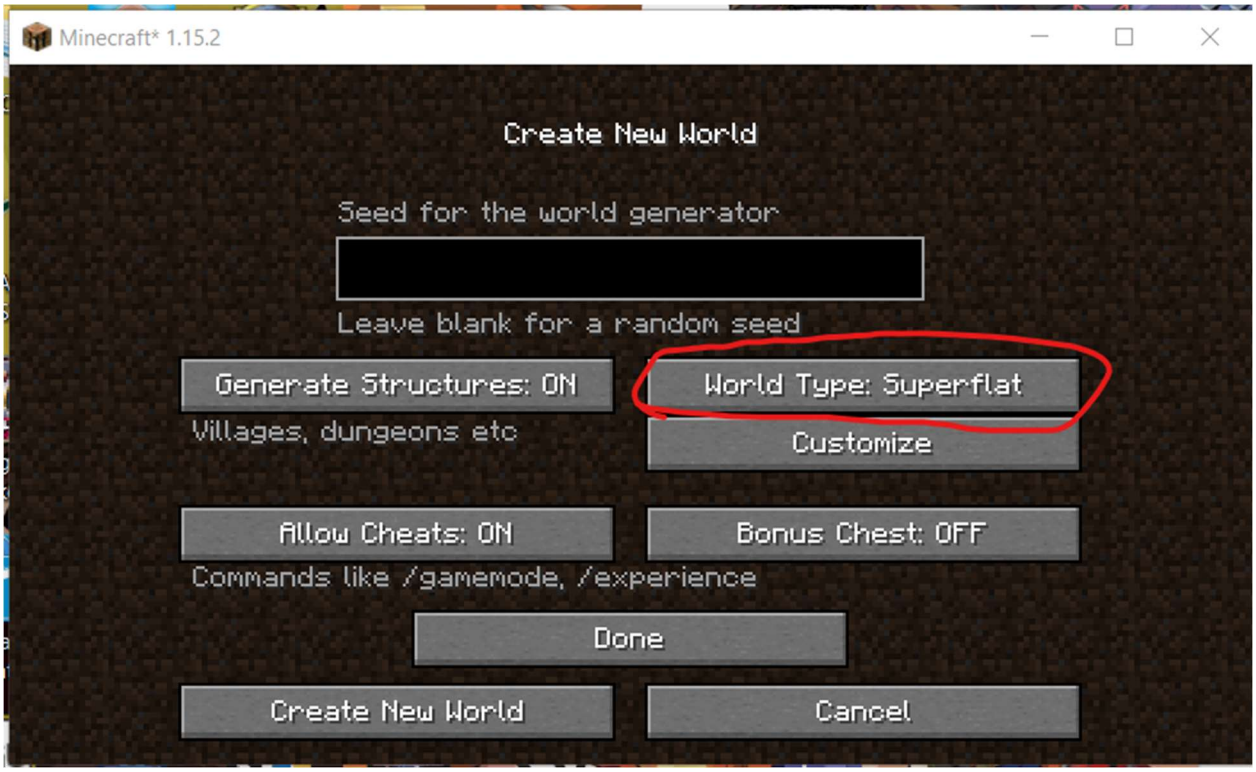
Then I selected Create New World



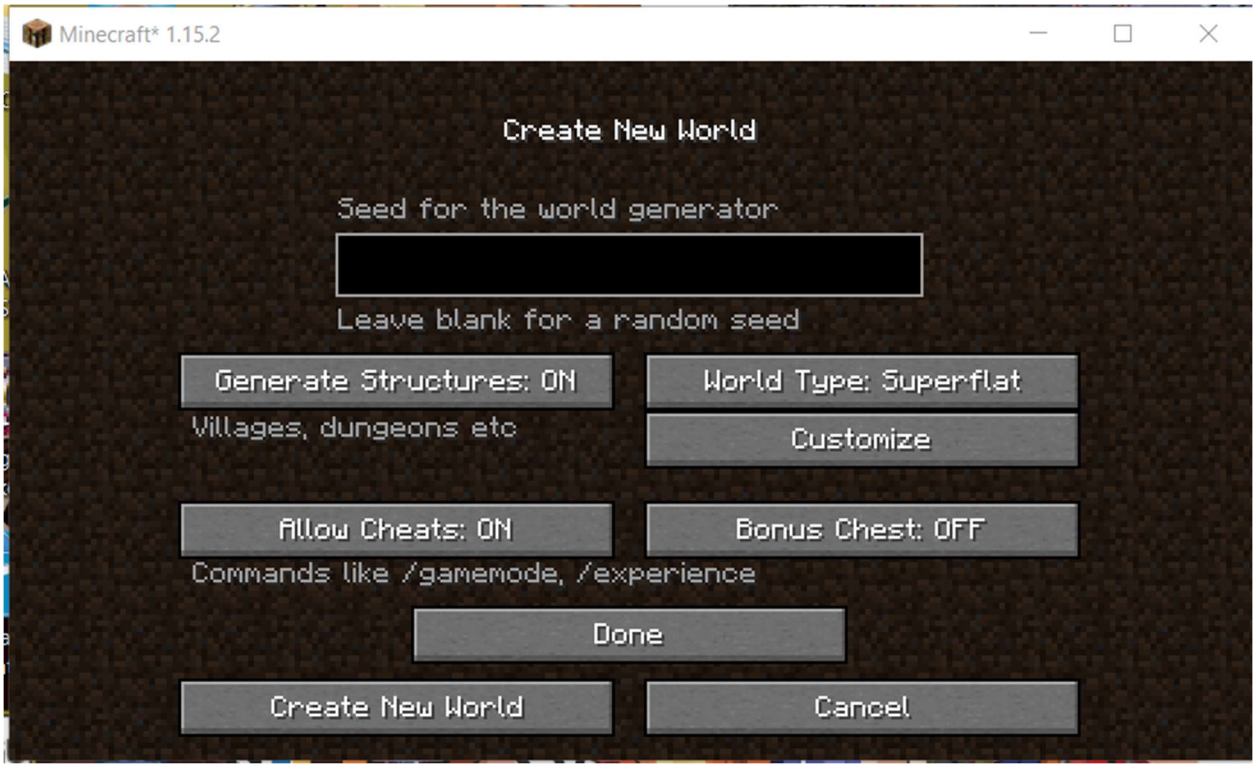
I selected Game Mode: Creative



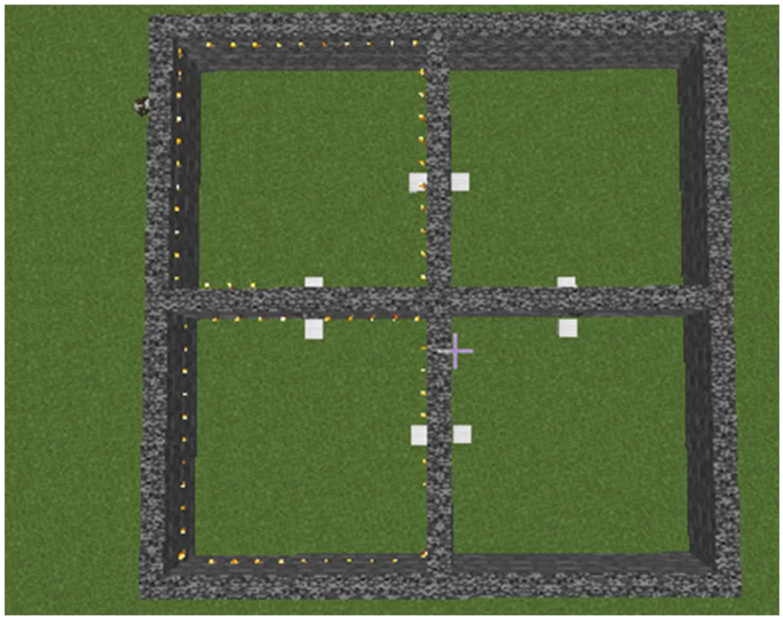
I choose more world options and selected superflat world



Then create new world



Next was deciding the area dimensions. I went with 25x25 and 4 high. And divided the room into 4 smaller rooms



This is the result of my work, the idea was to have “room” that the player would navigate through, but never got that far.