Minecraft U Sequence 4:

qCraft is a mod that brings the principles of quantum physics to the world of Minecraft. “Mods”, or game modifications, are programs that change the game in any way the mod developer desires. Campers will learn about the special objects and properties of qCraft, then learn just enough Java to get dangerous within a mod development environment and learn what it takes to become a game developer.

qCraft

qCraft is a mod that allows us to use principles of quantum physics.
All items can be found in creative mode. Recipes can be found at the qCraft wiki:
https://sites.google.com/a/elinemedia.com/qcraft/wiki/qcraft/blocks-and-items

3 Main Principles

The 3 principles you will learn to use are:

1. Observer Dependence
2. Superposition
3. Entanglement

You may also learn to utilize quantum teleportation.

Helpful video here.

Observer Dependence

Observational Dependency essentially states that looking at an object can actually change it.

In Minecraft, imagine a block that is diamond if you observe it from the North, but is coal from the West.

Superposition

Superposition is a state that includes all possible states.

For example, if I rolled a dice then covered it before it stopped, you wouldn't know if its a 1 or a 6 or a number in between, but you know it is *one* of those. However, on a quantum level, the dice would be all *six* possible outcomes at the *same* time until you looked.

In Minecraft, imagine a block that is both diamond and coal. Looking at that block will cause it to become one of the two, but you can't know which one until you look at it.

Entanglement

Entanglement refers to the ability of pairs or groups of particles that are created together to have a special relationship: their states are related. Not necessarilly the same state, but changing the state of one particle will have an effect on all its entangled partners. What is incredible though, is that this change occurs instantaneously, regardless of the distance seperating the particles. This means that the particles can exchange information faster than the speed of light!

Some believe this principle could lead to breakthroughs in quantum *teleporatation*.

In Minecraft, imagine having an entangled pair of diamond/coal superposition blocks. If we observe one of them and it is diamond, its partner, regardless of the distance seperating the two, will also be diamond.

Note: You can only entangle qCraft blocks.

New Items

Quantum ore/dust

1. Essence of Observation (EoO)
2. Essence of Superposition (EoS)
3. Essence of Entanglement (EoE)
4. Automated Observer (AO)
5. Quantum Computer

Practical Uses

Teleporter/Quantizer

qCraft introduces a new item called a “Quantum Computer” (QC) and it is mainly used to create a teleporter/quantizer.
A quantizer will “digitize” a part of the Minecraft world which can then be copied (quantized) or transported (teleported). The only difference is that a teleporter recquires an *entangled pair of quantum computers*.

Items Needed:

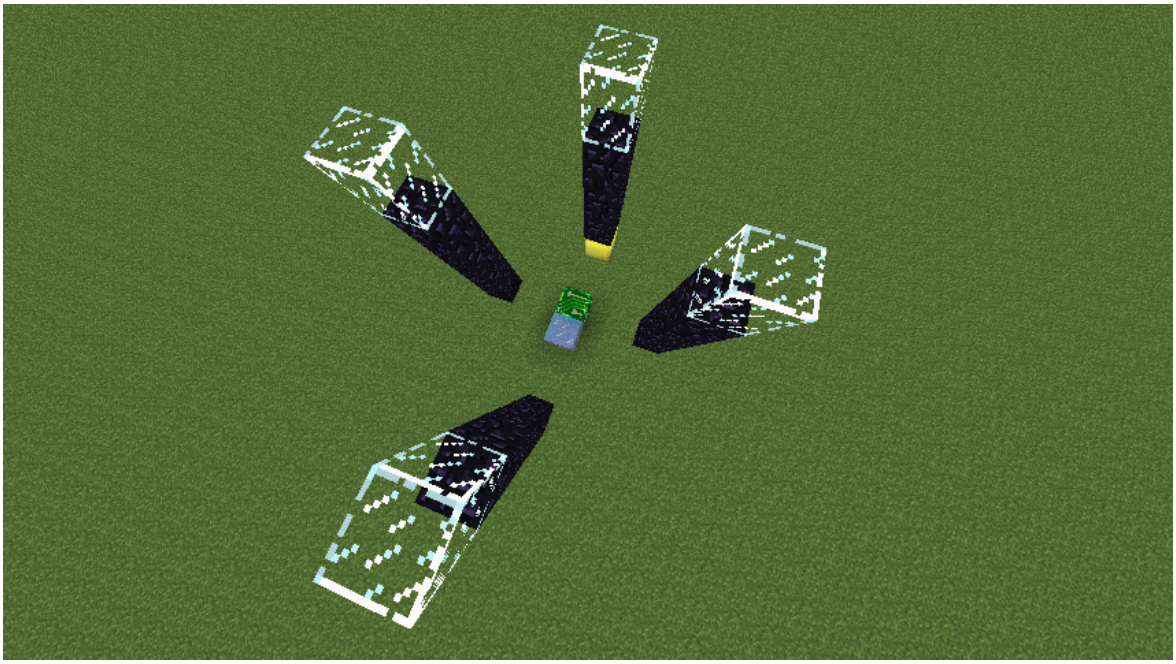
- Obsidian
- Full gold blocks
- Ice
- Glass blocks
- EoO
- Quantum Computer (optional entangled pair)

Crafting ODB’s

ODB recipe. \label{fig:ODB recipe}

1. Craft 4 different Observational dependent blocks (ODB's) using Gold for one of the cardinal directions, and Obsidian for the rest.
2. Place these ODB's in a square/rectangle around a Quantum Computer (QC) such that all the ODB's resolve to Gold when you are standing in the enclosed area.
3. Put a block of Ice next to the QC to provide cooling.
4. Place Obsidian blocks on top of the ODB's, as high as you want your teleportation area to be (These are called "Pylons").
5. Then place a glass block on top of your Pylons.
6. Construct another area in exactly the same way and **SAME DIMENSIONS**
7. "Energize" the QC

Your teleporting Matrix should look like this:



An example of a teleporting matrix. \label{fig:telporterMatrix}

Note: This will not transport you the player, only items/blocks inside the area. **There is no distance cap!!!**

If you used entangled QC's then the contents of the two teleporters will be swapped. If you chose to build a quantizer instead, break the Quantum computer and walk/run/swim/tp to the destination area, place the QC and activate it again.

Troubleshooting:

1. The QC / pylon matrix dimensions don't exactly match at origin and destination
2. There is no room for quantized blocks at destination. (e.g. trying to move a tall tower to a mountaintop near the height limit of the world)
3. Destination is abutting water or lava that could flow in and damage structures
4. Source / destination contains blocks with extra info that would be lost in the transfer (e.g. blocks from other mods with unusual data storage methods)

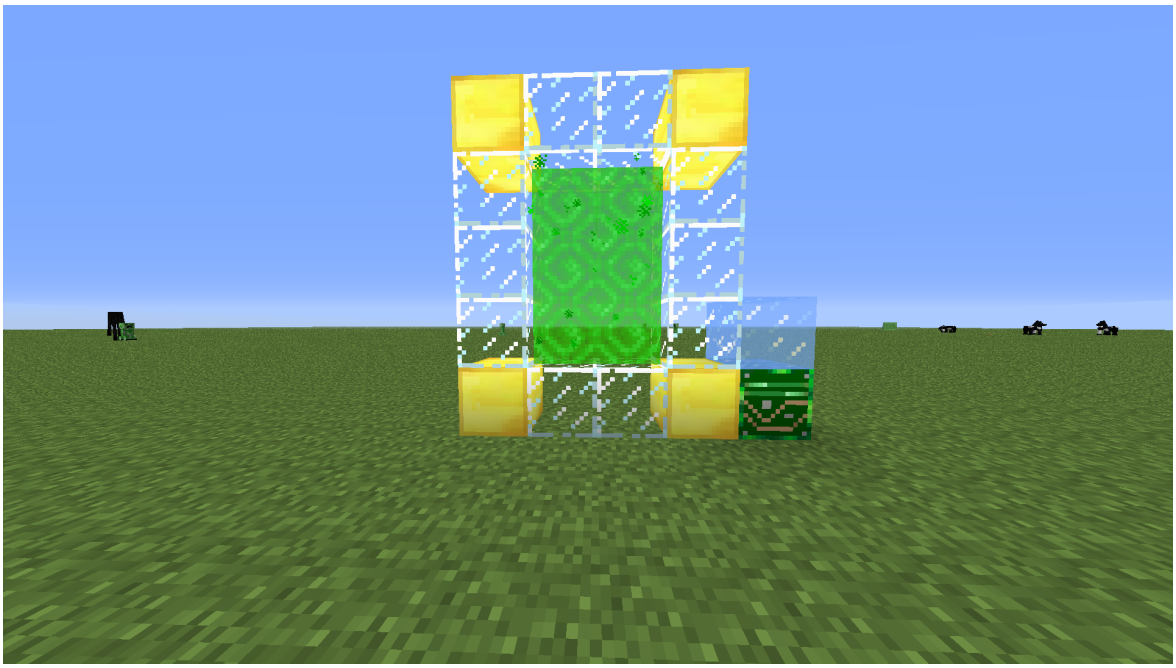
Quantum Portal

If you were disappointed that the Teleporter didn't take you with the rest of your stuff, don't worry, qCraft has another great feature that will let you travel anywhere, including between servers.

You will need the same items for the portal as the teleporter/quantizer:

- Obsidian
- Full gold blocks
- Ice
- Glass blocks
- EoO
- Quantum Computer
- Create gold/obsidian ODB's such that they will resolve to Gold only when you try entering the Portal (so opposite cardinal directions, i.e. North and South)
- Create the same structure as a nether portal only use the gold/obsidian ODB's for the corners and glass in between.
- Place a quantum computer and a block of ice next to your portal

It will look like this



An example of a Quantum Portal. \label{fig:portal}

When you right-click the QC you will see this menu:

QP Menu. \label{fig:menu}

Name the Portal you have selected and give it a destination portal. You will need to build at least two portals to get anywhere.

Controlling Lighting in a (large) room

We are not limited to *entangling* merely two (2) objects together, we can in fact entangle a multitude together. All you have to do is use one of the previously entangled blocks in the recipe. This allows us to have a mass number of objects react to a single action. For example, I could entangle several superpositional redstone lamps together, and light them all at once simply by lighting one.

Make an adventure map using the features of Minecraft

Nifty Tricks

These are some ways in which you can use some of the cool properties of qCraft blocks to create a new, fascinating type of adventure map.

Utilizing Invisibility

What you may not know, is that leaving one of the directional blocks empty when crafting ODB's or QB's will make them *invisible* and permeable from that direction or instance. You can combine this with the properties of entangled groups to create trapdoors that activate upon entering a room, disappearing staircases, one way bridges, and hallways that weren't there a second ago.

Portal loops

Think of an elevator to a building. Every portal you build is a door that elevator can stop at. However, the floor you stop at will decide what your next stop will be. You can use this to create a progression of portals to take players to different areas/levels.

Examples:

Fixed Path:
1. Portal 1 -> Portal 2
2. Portal 2 -> Portal 1

Progression
1. Portal 1 -> Portal 2
2. Portal 2 -> Portal 3
3. Portal 3 -> Portal 4
4. etc…

Converging
1. Portal 1 -> Portal 3
2. Portal 2 -> Portal 3
3. Portal 4 -> Portal 3

Implementing New Redstone: Automated Observer

qCraft features a new redstone item called an Automated Observer (AO). An AO must be placed adjacent to the block it is observing. The AO will only observe one block and will only do so as long as

there is a redstone powering it. You can combine this with logic gates and the invisibility property to make some interesting puzzles.

New Items: Goggles

Another cool set of items in qCraft are goggles: Quantum and Anti-Observation

Quantum Goggles: When worn, you will see all quantum blocks in their superpositional state (green and swirly). This makes it relatively easy to differentiate quantum blocks from regular blocks and will also reveal invisible quantum blocks.

Anti-Observation Goggles: When worn, these goggles will make quantum blocks act as though your character wasn't there. They will then resolve into their last observed state. This can be used to work around or with ODB's and QB's without them constantly changing.

Entanglement Explained

If you tried being creative, you're mostly likely all tangled up trying to entangle things. Heh. Not to worry, this should help you become *untangled*. Okay, enough jokes.

1. You can ony entangle ODB/quantum blocks.
2. You can make quantum blocks and ODB's with the same block from all sides.
3. Quantum blocks made from multiple types of blocks will change properties based on what state they resolve to.
4. Large entangled groups are made by entangling a new ODB with an ODB already in the entangled group.
5. You cannot un-entangle anything (without destroying it).
6. Entangled superpositional blocks will all resolve to the same block, making them more dependable than ODB's in large groups.

Mod Dev

Setup

1. Extract the “forge” folder.
2. Shift + right-click inside the folder and choose “Open command window here”. A console window will appear.
3. Type `gradlew setupDecompWorkspace --refresh-dependencies` and press `Enter`. This will download the Minecraft source code and decompile it so we can work with it to make our mods. If any errors occur, the program should give you suggestions on what to fix before trying again.
4. Type `gradlew idea` and press `Enter`, which sets up an IntelliJ workspace for coding.
5. Open IntelliJ and select the created project at the top.

Creating new blocks

> In this section, be sure to match ALL capitalization and punctuation as exactly as possible.

Programmers use something called a *class* as a blueprint for all of the objects in Minecraft. There’s a class for a diamond pickaxe, for an iron ore, and all other blocks and items. By creating our own class, we can add our own objects to the game. Rather than make a completely new class, however, we can *extend* the existing Block class and make our own new block. It will have all the normal properties of a block (can be broken, placed down, picked up) but we can set our own texture, hardness, and sound.

Create a new class called “GenericBlock” by right-clicking on the package and choosing “Create new class”. Name it “GenericBlock” and press OK. The following code should be displayed. You should add the lines with comments after them.

```
package com.example.examplemod;

import net.minecraft.block.Block; //Add me

/**
 * Created by atvaccaro on 6/21/2014.
 */
public class GenericBlock extends Block //Add the second half of this line
{
}
```

An error will come up regarding a missing *constructor*. Constructors are a set of functions that run whenever a new block is created in the game. The following constructor should go inside the braces of the GenericBlock class. If your cursor is over `GenericBlock`, you should be able to press Alt-Enter and select “Add missing constructor”.

```
protected GenericBlock(Material p_i45394_1_)
{
    super(p_i45394_1_);

    setHardness(0.5F);
    setStepSound(Block.soundTypeGravel);
    setBlockName("genericDirt");
    setCreativeTab(CreativeTabs.tabBlock);
    setBlockTextureName("examplemod:genericDirt");
    setHarvestLevel("shovel", 0); //0 is wood
}
```

Taking a look at our method calls, it will be made of the “ground” material, sound like gravel, can be broken more easily by shovels, and be called “genericDirt” among other properties.

However, simply making a new class is not enough. To actually add our block into the game, we need to register it with Minecraft Forge. The following block of code goes into the ExampleMod.java class, which you can open from the project tree on the left side of the screen.

```
@EventHandler
public void preInit(FMLPreInitializationEvent event)
{
  Block genericDirt = new GenericBlock(Material.ground).
  GameRegistry.registerBlock(genericDirt, "genericDirt");
}
```

This block of code registers our newly-created block with the game.

So to recap, the ExampleMod.java and GenericBlock.java files should look as follows.

```java
// ExampleMod.java
package com.example.examplemod;

import cpw.mods.fml.common.registry.GameRegistry;
import net.minecraft.block.Block;
import net.minecraft.block.material.Material;
import net.minecraft.creativetab.CreativeTabs;
import net.minecraft.init.Blocks;
import cpw.mods.fml.common.Mod;
import cpw.mods.fml.common.Mod.EventHandler;
import cpw.mods.fml.common.event.FMLPreInitializationEvent;
import cpw.mods.fml.common.event.FMLInitializationEvent;
import net.minecraftforge.common.MinecraftForge;

@Mod(modid = ExampleMod.MODID, version = ExampleMod.VERSION)
public class ExampleMod
{
    public static final String MODID = "examplemod";
    public static final String VERSION = "1.0";

    @EventHandler
    public void preInit(FMLPreInitializationEvent event)
    {
        Block genericDirt = new GenericBlock(Material.ground);
        GameRegistry.registerBlock(genericDirt, "genericDirt");
    }

    @EventHandler
    public void init(FMLInitializationEvent event)
    {
                // some example code
        System.out.println("DIRT BLOCK >> "+Blocks.dirt.getUnlocalizedName());
    }
}
```

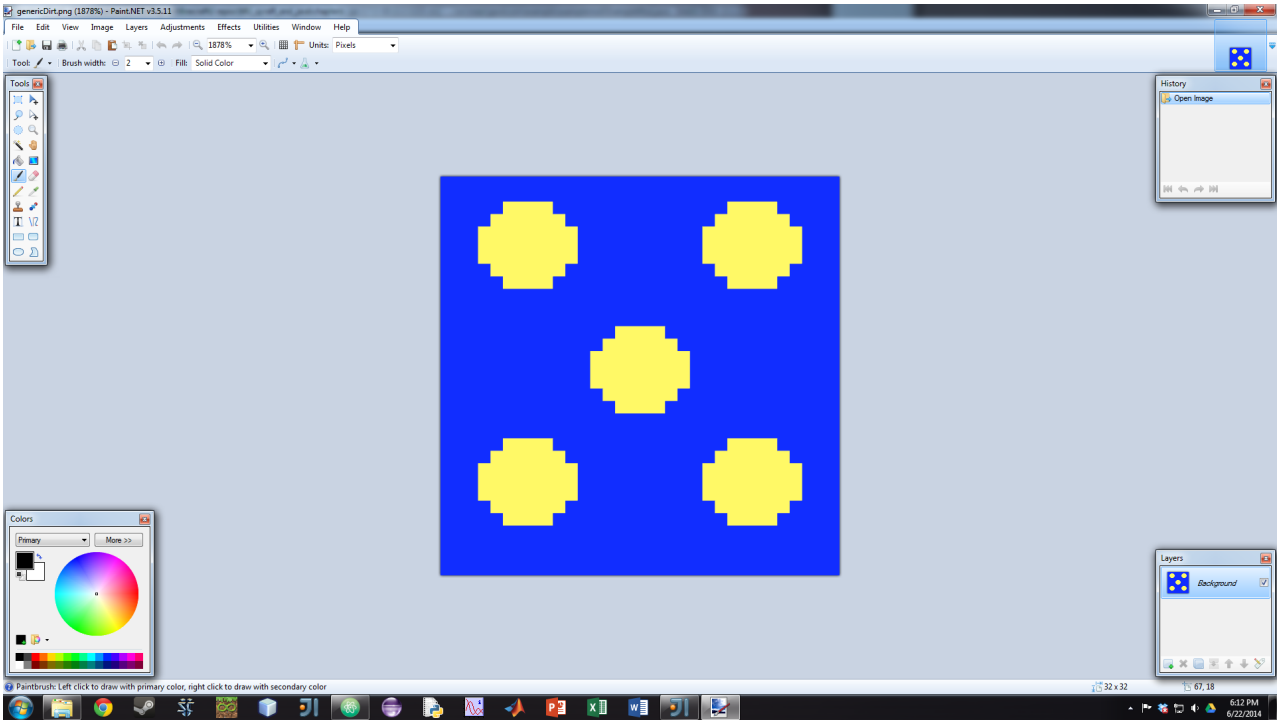
```java
package com.example.examplemod;

import net.minecraft.block.Block;
import net.minecraft.block.material.Material;

/**
 * Created by atvaccaro on 6/21/2014.
 */
public class GenericBlock extends Block
{
    protected GenericBlock(Material p_i45394_1_) {
        super(p_i45394_1_);

        setHardness(0.5F);
        setStepSound(Block.soundTypeGravel);
        setBlockName("genericDirt");
        setCreativeTab(CreativeTabs.tabBlock);
        setBlockTextureName("examplemod:genericDirt");
        setHarvestLevel("shovel", 0); //0 is wood
    }
}
```

To actually launch our modded Minecraft, run the project by clicking on the green arrow at the top of the IDE. Make a new creative world and try placing your block on the ground. It should be under the normal “blocks” tab at the very bottom and will be called *tile.genericBlock.name*. The coloring should be a purple and black checkerboard, the default color scheme when a texture is not specified.

The custom block we have just added.

Adding textures to blocks

1. First, create the folder that will hold our textures. The full path is “src/main/resources/assets/examplemod/textures/blocks”. Start at the “resources” folder (it should already exist) and create one folder after another until you get down to the final one, “blocks”.
2. Open up Paint and create a new empty canvas. The resolution should be square; most Minecraft textures are 16x16 but you could also try 32x32 or 64x64.
3. Take a few minutes and make your own texture! For right now, the block will have the same texture on all six sides like cobblestone or obsidian. Save the texture as .png called “genericDirt” in the “blocks” folder. I made a dice-like yellow and blue texture for mine.

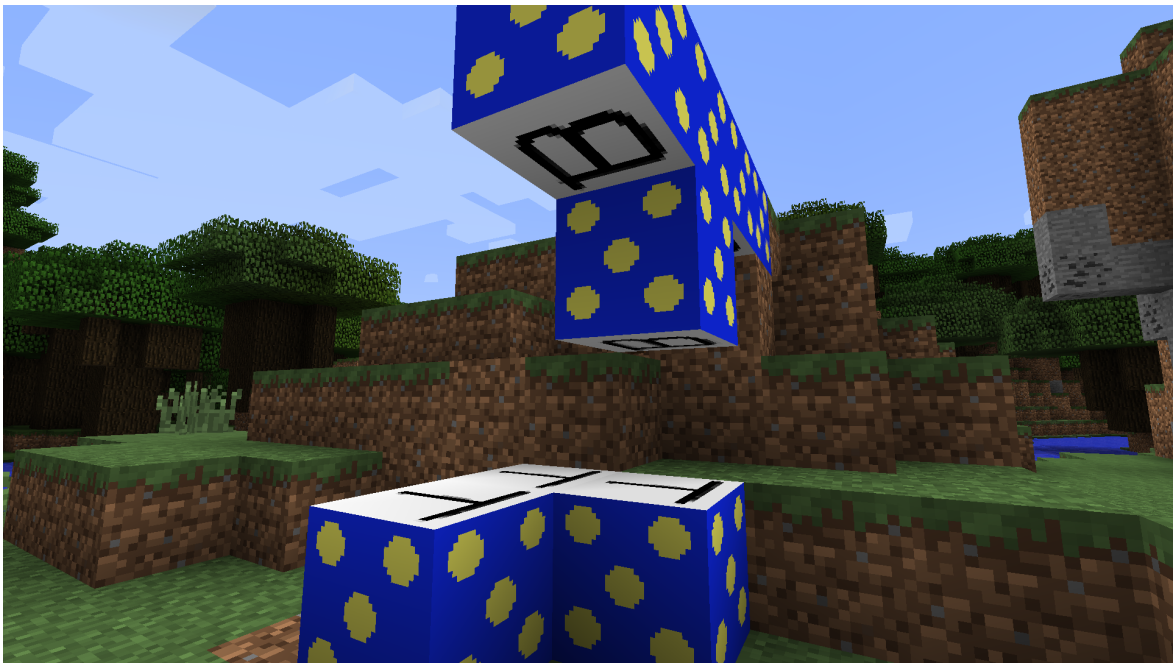


4. After your texture has been saved, run Minecraft. Now check out the texture of your block!

Multi-sided Textures

What about blocks like grass that have different textures on different sides? The following code in the GenericBlock class registers several different textures and tells the program which textures belong on which side.

```java
//Creating our icon variables
private IIcon topIcon;
private IIcon sideIcon;
private IIcon botIcon;

//Registering specific textures with each icon variable
@Override
public void registerBlockIcons(IIconRegister par1IconRegister)
{
    this.topIcon = par1IconRegister.registerIcon("examplemod:genericDirtTop");
    this.sideIcon = par1IconRegister.registerIcon("examplemod:genericDirt");
    this.botIcon = par1IconRegister.registerIcon("examplemod:genericDirtBot");
}

//This function tells which icons to use for which sides of the block
@Override
public IIcon getIcon(int par1, int par2) //par1 = the side of the block
{
    if (par1 == 0) //Bottom
        return botIcon;

    else if (par1 == 1) //Top
        return topIcon;

    else //all other sides
        return sideIcon;
}
```

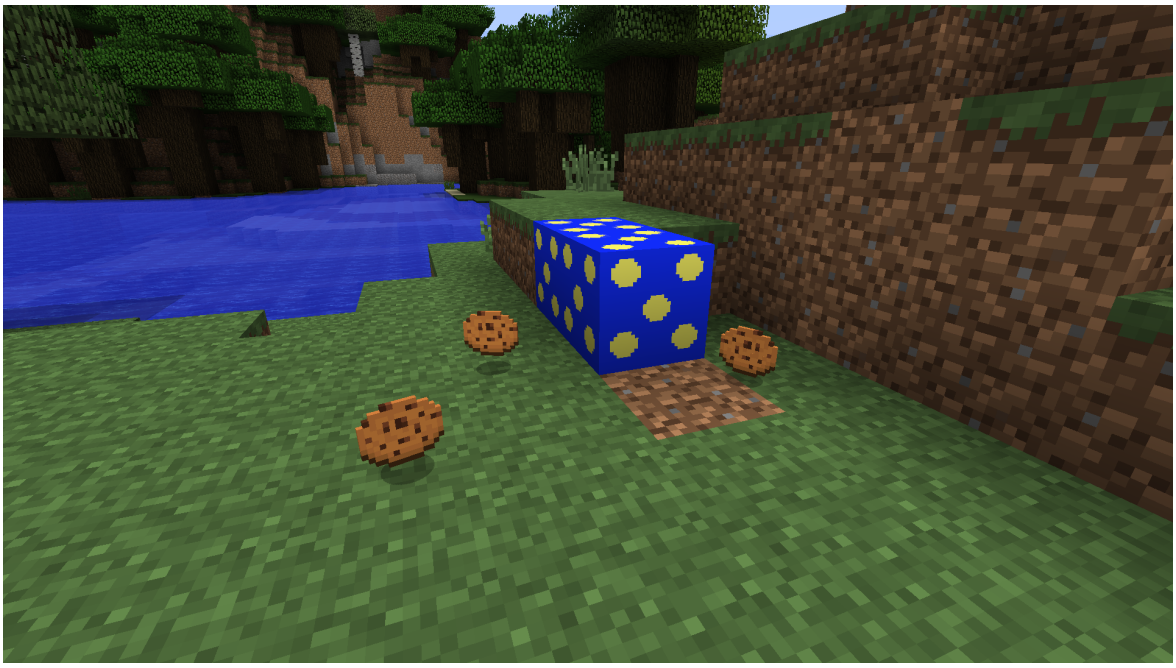


Our block with multiple textures.

Specifying dropped items

Right now, our custom block drops itself just as dirt and wood does. But what about glowstone? It drops an item (or items) when broken. We can modify our existing block easily to drop an item as well. The following function declaration goes into our GenericBlock class.

```
@Override
public Item getItemDropped(int metadata, Random random, int fortune)
{
    return Item.getItemById(357); //What item has an ID of 357?
}
```



Our block now drops cookies.

Mod Dev

Making new crafting recipes

There are two types of recipes we can create as well: shapeless and shaped. Shapeless recipes (such as making wooden planks from wood) don't require the items to be in any specific orientation to work. Shaped recipes (such as making tools) requires blocks to be in specific locations for the recipe to function. You will also need to import the ItemStack and Items classes to use these calls (IntelliJ will prompt you for them when you type your functions).

Important!

These crafting and smelting recipes go in our original ExampleMod class.

Steps

First, let's make a shapeless recipe.

```
GameRegistry.addShapelessRecipe(new ItemStack(Items.diamond, 64), new ItemStack(Blocks.
```

This recipe simply trades in a stack of 64 dirt blocks for 64 diamonds.

A dirt-for-diamonds recipe.

To add more items, simply add more ItemStacks within the parentheses.

```
GameRegistry.addShapelessRecipe(new ItemStack(Items.diamond, 64), new ItemStack(Blocks.
    new ItemStack(Blocks.dirt), new ItemStack(Blocks.dirt));
```

A recipe that requires 3 dirt per diamond.

Next, let's make a shaped recipe. Shaped recipes group the rows of items using letters to represent the type of block. For example, the recipe for TNT would be:

```
("xyx", "yxy", "xyx", 'x', new ItemStack(Materials.sulphur), 'y', new ItemStack(Blocks.sand))
```

Pay attention to the double-quotes for the letters representing the recipe shape and the single-quotes for the letters representing the items in the recipe.

> Note: when we leave out the size of the ItemStack, the game will assume a size of 1.

An example of a shaped recipe. You use three strings to represent the three rows of the crafting table and define which letters relate to which type of item. Spaces are used to represent empty crafting slots.

```
GameRegistry.addShapedRecipe(new ItemStack(Items.diamond), "xxx", "x x", "xxx", 'x',
    new ItemStack(Items.coal));
```

The crafting diamond recipe.

Smelting recipes

Smelting recipes are very similar and only require a slightly different function call.

```
GameRegistry.addSmelting(Blocks.stone, new ItemStack(Blocks.stonebrick), 0.1f);
```

This time, the input item is on the left while the output is on the right. The number at the end specifies how much experience the player receives from the smelting.

The smelting diamond recipe.

On a side note, the *damage values* of items often hold *metadata* about the block. For example, all the colors of wool are actually the same type of block. They're rendered differently based on the value of their damage value. We can use this data value in our recipes to alter what kinds of wool, clay, or wood are required.

```
// What do you think this smelting recipe does?
ItemStack woolStackBlack = new ItemStack(Blocks.wool);
ItemStack woolStackOrange = new ItemStack(Blocks.wool);
woolStackBlack.setItemDamage(15);
woolStackOrange.setItemDamage(1);
GameRegistry.addSmelting(woolStackBlack, woolStackOrange, 0.1f);
```

The smelting wool recipe.

Setting names

The names that we've given our new blocks and items so far are all hard-coded into our mod. But what if we want people in other countries who speak different languages to play our mod? We can use what are called *language packs* to give our items language-specific names. The packs also replace the cumbersome "package.item.item"-type names with real names such as "Iron Ingot" or "Dirt".

File Extensions

Before we create our language packs, make sure you have "hide file extensions for known file types" disabled. To check if its disabled, go look at your textures. If they show up as "*name*.png" you're good. Otherwise, follow these instructions.

Windows

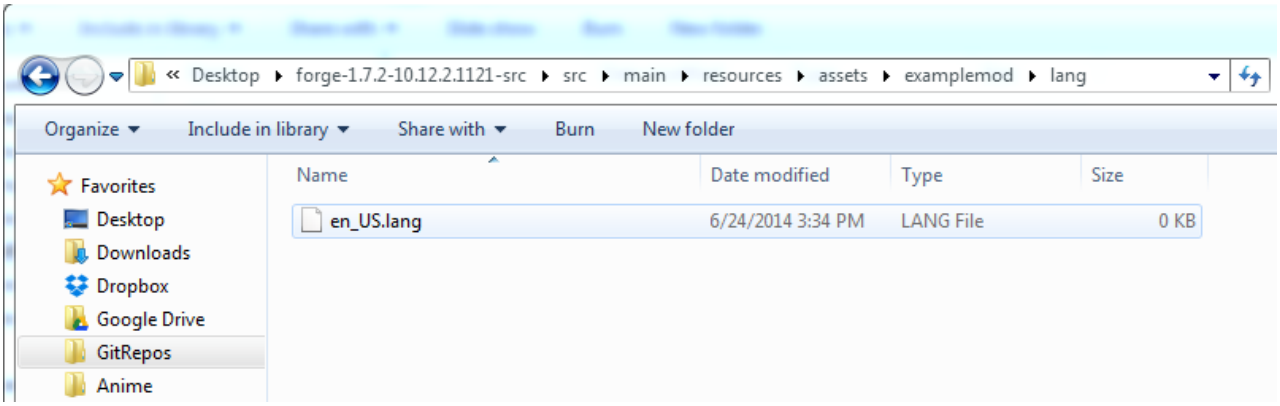
1. Start -> Control Panel -> Appearance and Personalization -> Folder Options
2. Click on “View” tab
3. Click “Advanced settings”
4. Uncheck the box next to “Hide extensions for known file types” then click “OK”

Mac

1. Select Finder -> Preferences -> Advanced
2. Select “Show all filename extensions”

Language Packs

1. Create a folder called “lang” in the “assets/examplemod” folder.
2. Create a new text file called “en_US.lang” in the folder.



3. Right-click on the file and choose to edit it. The default entry is something similar to: `category.blockName.name=Item Name`. The name that appeared for our initial block, `tile.genericDirt.name` is what we would use in our case. So my line would be `tile.genericDirt.name=CookieDice`.
4. Save the file and run Minecraft. Your name should now appear in-game.

CookieDice
64
61
13
10
8
5

Appendix 1: MCU Servers

We have a lot of cool stuff on the servers and have more things planned, like build projects, adventures and online lessons. Right now there is a Minecraft U building with all the redstone logic gates we built in our redstone camps; there are two large mazes built with ComputerCraft turtles and maze making algorithms; an obstacle race course; a distant and mysterious pagoda that whispers to you as you approach; a special ComputerCraft island and an undersea village.

And the main point of maintaining the server—you know who is on it. Only campers and staff, plus a very few vetted outside players, are on the MCU servers.

This is all included as a part of your week at Minecraft U, and you may continue to play on the server indefinitely.

Access to the camp server is contingent on behavior guidelines just like those in real life. Anything you wouldn't do to someone else in real life, you should not do on the server. PVP is turned off. Do not ask for it to be turned on.

Violation of any of these rules will result in bans on a graduated scale described below.

1. No griefers. A griefer is a player in a multiplayer video game who deliberately irritates and harasses other players within the game. This includes causing damage to other players, damaging existing structures/contraptions and stealing.
2. It bears repeating: No stealing. You wouldn't just walk into someone else's house and take things out of their closet. Similarly, you may not remove things from other player's chests or houses without consequences. The only exceptions to this are chests marked with a sign as community chests, and food in the case of emergency.
3. No bad language.
4. No raging or rage quitting. Even if not intended as raging, all caps chatting may be interpreted as such.
5. Fill creeper holes and repair any other damage done by creeper blasts. (We have disabled creeper block damange on the servers for now.)
6. Replant community crops, re-breed community animals and leave as much behind in community chests as you take out.

Punishment for breaking any of the server rules are as follows:

1. Verbal warning through chat.
2. If immediately reachable, warning to a parent.
3. Kick.
4. Indefinite ban. Length determined after discussion with a parent.
5. Permanent ban. Reversible only through appeal.