Custom Block rendering Tutorial A tutorial by riccardoNL

Posted 16 November 2011 - 05:40 PM

Hey everyone! Alot of poeple asked how you can make a custom block render, like a torch or a fence. Well, I know how to, so that's why I made the tutorial.

TO MOD TUTORIAL CREATORS: It is OK to copy the tutorial to a modding forum or your own tutorials ONLY IF YOU PLACE MY NAME UNDERNEATH IT AND A LINK TO THIS POST!!!

Thank you

IF YOU NEED HELP, Place a reaction + PM me saying you placed a reaction, else, I am NOT notified if you place a reaction, and i probably don't see it..

Ok, i'll explain how to make a custom block render of the block 'TutorialBlock' (Note that I wrote all of this, and if you want to make a custom block render, you can't just copy everything, play around with it!)

I assume you know that i = X, j = y, k = Z...

First, let's make a mod_ file (Called mod_TutorialBlock in this case)

Start of the file like this:

```
package net.minecraft.src;

import java.util.*; //This imports all java, it doesn't have to, but just to be sure it works in this tutorial

public class mod_TutorialBlock extends BaseMod
```

And open the file with a { bracket (don't forget!!)

and above the constructor, create a block (which will be extended in the 'static'), a custom blockrender ID, and a custom texture name.

```
public static Block TutorialBlock;
public static int tutorialblockRenderID;
public static int TutorialBlockTexture;
```

Now, for the constructor, register the block and add a name. Also, register the custom block render.

```
public mod_TutorialBlock()
{
ModLoader.AddName(TutorialBlock, "Tutorial Block by riccardoNL");
ModLoader.RegisterBlock(TutorialBlock);
tutorialblockRenderID = ModLoader.getUniqueBlockModelID(this, false);
}
```

Underneath that, place the static. Create the block, and the texture for it.

```
static
{
TutorialBlockTexture = ModLoader.getUniqueSpriteIndex("/terrain.png");
ModLoader.addOverride("/terrain.png", "/tutorialblock.png", TutorialBlockTexture);
TutorialBlock = (new BlockTutorialBlock(200, TutorialBlockTexture)).setHardness(1.0F).setResistance(5000.0F).setBlockName("TutorialBlock");
//note that the 200 = the block ID, and the TutorialBlockTexture after it is the icon it shows ingame, in your inventory
}
```

Now, lets make a function for the Custom Block renders:

```
public boolean RenderWorldBlock(RenderBlocks renderblocks, IBlockAccess iblockaccess, int i, int j, int k, Block block, int l)
{
if (l == tutorialblockRenderID)
{
return RenderTutorialBlock(block, i, j, k, renderblocks);
}
return false;
}
```

If you want to have more than one blockrenders in a file, do it like this:

```
public boolean RenderWorldBlock(RenderBlocks renderblocks, IBlockAccess iblockaccess, int i, int j, int k, Block block, int l)
{
if (l == tutorialblockRenderID)
{
return RenderTutorialBlock(block, i, j, k, renderblocks);
}
if (l == otherblockRenderID)
{
return RenderTutorialBlock(block, i, j, k, renderblocks);
}
return false;
}
```

Etc..

Now, here comes the part that is hard for most people because they don't understand where all the things stand for.

For every block render in the file you have, you have to make a new Public boolean RenderWorldBlock. In this tutorial, i only make one block.

Let's start with the boolean:

```
public boolean RenderTutorialBlock(Block block, int i, int j, int k, RenderBlocks renderblocks)
{[/size][/size][/size]
[size="2"][size="2"][size="2"]}
```

Now, between the brackets, you have to define your block render. Let's start from the bottom, since that piece of the code should always be there:

```
block.setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 1.0F, 1.0F);
return true;
[/size][/size][size=2]
```

These numbers stand for the FULL block bounds. The return true; stands for the boolean to actually apply the render.

Let's take a look at the blockbounds of the FULL block;

(0.0F, 0.0F, 0.0F, 1.0F, 1.0F, 1.0F)
What do these nubers stand for? Well, there is an I, a J, and a K (x, y z,) where the block STARTS (the first 3 numbers), and there are 3 I, J and K's where the block ENDS.
This is a block that is square, and has the same form as all the other blocks in minecraft (think of stone, dirt, etc). The block starts at 0, and ends with 1, at every axis. So it is a square. The first and 4th nuber stands for the X (left and right), the second and 5th stand for the Y (up and down) and the third and 6th stand for the Z (depth) Let's take a look at the code for a half slab, which is only 0,5 blocks high:

(0.0F, 0.0F, 0.0F, 1.0F, 0.5F, 1.0F)

See, the Y starts at 0 9the bottom of the block) and ends with 0.5 (a half block). Starting to get it?

Well, if you don't, just read the entire piece over and over again..

Now, this is only the TOTAl block's edges, the bounding box. What if we want smaller blocks INSIDE the block?

```
renderblocks.overrideBlockTexture = TutorialBlockTexture;
block.setBlockBounds(0.2F, 0.2F, 0.2F, 0.8F, 0.8F, 8.0F);
renderblocks.renderStandardBlock(block, i, j, k);
renderblocks.overrideBlockTexture = -1;
```

For every box you want INSIDE the total block, you will need this above piece. You CAN have multiple textures for different parts of the block, just make another texture with a variable.

This smaller block starts at 0.2 at all axis and ends at 0.8. So this is a cube ass well, all sides are the same length.
This part: renderblocks.overrideBlockTexture = -1; is very important, else, the texture from this block will spread and your enviroment will get that texture 😋

Let's make a straight torch with a lightbulb.

```
public boolean RenderTutorialBlock(Block block, int i, int j, int k, RenderBlocks renderblocks)
{
renderblocks.overrideBlockTexture = [/size][/size][/size][/size][size=2]TutorialBlockTexture[/size][size="2"][size="2"][size="2"][size="2"];
block.setBlockBounds(0.4F, 0.0F, 0.4F, 0.6F, 0.5F, 0.6F);
renderblocks.renderStandardBlock(block, i, j, k);
renderblocks.overrideBlockTexture = -1;

renderblocks.overrideBlockTexture = [/size][/size][/size][/size][size=2]TutorialBlockTexture[/size][size="2"][size="2"][size="2"][size="2"];
block.setBlockBounds(0.2F, 0.5F, 0.2F, 0.8F, 1.0F, 0.8F);
renderblocks.renderStandardBlock(block, i, j, k);
renderblocks.overrideBlockTexture = -1;

block.setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 1.0F, 1.0F);
return true;
}
```

This will create a pole with a cube on top. Let's take al look at the code for the pole:

```
block.setBlockBounds(0.4F, 0.0F, 0.4F, 0.6F, 0.5F, 0.6F);
```

X and Z start at 0.4 and go to 0.6. so it's a 2 X 2 square, at the middle of the block. the Y goes from 0 to 0,5, that is the height. With this, you have a half block high pole.

The cube:

```
block.setBlockBounds(0.2F, 0.5F, 0.2F, 0.8F, 1.0F, 0.8F);
```

X and Z start at 0.2 and go to 0.8, a 6 X 6 qsuare, in the middle of the block, too. But it is above the pole, because the Y starts at 0.5(where the pole ended) and ends at 1.0, where the complete block ends.

You can have as many of these little blocks in your FULL block, but keep in mind stat they can't cross each other...

After the render of the block, the string version:

```
public String Version()
{
return "Tutorial by riccardoNL";
}
```

and end the file with a closing } bracket. And now. Done? No, the render works, but we still have to tell Minecraft to render the block like that. You can do that in the block file. Let's see:

```
package net.minecraft.src;

import java.util.Random;

public class BlockTutorialBlock extends Block
```

Basic file start, open with a { bracket (!)

The constructor:

```
public BlockTutorialBlock(int i, int j)
{
super(i, Material.wood);
blockIndexInTexture = j;
setStepSound(soundWoodFootstep);
}
```

And after that, place this to tell the game that you can 'see' through some parts of the block (like a fence, or a torch, theyare not completely square)

```
public boolean isOpaqueCube()
{
return false;
}
```

And tell it to render the block differently than normal:

```
public boolean renderAsNormalBlock()
{
return false;
}
```

Finally, tell the game to render the block like this:

```
public int getRenderType()
{
return mod_TutorialBlock.tutorialblockRenderID;
}
```

And end the file with a bracket. (!!!)

Done!! Place the textures in the jar, check if everything is ok, recompile and test it out! If it works, you can experiment with the numbers and add more blocks in your full block.

Hope i helped you!! If so, please click the green plus under my signature. Thank you!!

~riccardoNL