

```

package net.minecraft.src;

import java.io.IOException;

// Created by MoareAI
// Contains data handled by the ModLoader

public class mod_LogicalGates extends BaseMod
{
    //ModLoader actions
    public mod_LogicalGates()
    {
        RenderId1 = ModLoader.getUniqueBlockModelID(this, false);
        RenderGate = RenderId1;

        ModLoader.RegisterBlock(BlockNOT);
        ModLoader.RegisterBlock(BlockOR);
        ModLoader.RegisterBlock(BlockAND);
        ModLoader.RegisterBlock(BlockXOR);
        ModLoader.RegisterBlock(BlockNOR);
        ModLoader.RegisterBlock(BlockNAND);
        ModLoader.RegisterBlock(BlockXNOR);
        ModLoader.RegisterBlock(BlockDiode);

        //Block Texture
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGNOTOn.png",
TextureNOTOn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGNOTOff.png",
TextureNOTOff);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGOROn.png",
TextureOROn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGOROff.png",
TextureOROff);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGNOROn.png",
TextureNOROn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGNOROff.png",
TextureNOROff);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGDiodeOn.png",
TextureDiodeOn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/LGDiodeOff.png",
TextureDiodeOff);

        //Item Icons
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGNOT.png", IconNOT);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGOR.png", IconOR);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGAND.png", IconAND);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGXOR.png", IconXOR);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGNOR.png", IconNOR);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGNAND.png", IconNAND);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGXNOR.png", IconXNOR);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/LGDiode.png",
IconDiode);

        //Item and Block Names
        ModLoader.AddName(ItemNOT, "NOT Gate");
        ModLoader.AddName(ItemOR, "OR Gate");
        ModLoader.AddName(ItemAND, "AND Gate");
        ModLoader.AddName(ItemXOR, "XOR Gate");
        ModLoader.AddName(ItemNOR, "NOR Gate");
        ModLoader.AddName(ItemNAND, "NAND Gate");
        ModLoader.AddName(ItemXNOR, "XNOR Gate");
        ModLoader.AddName(ItemDiode, "Diode");

        //Recipe for Diode

```

mod_LogicalGates.java

```
ModLoader.AddRecipe(new ItemStack(ItemDiode, 6), new Object[] {
    "#X#", Character.valueOf('#'), Item.redstone, Character.valueOf('X'),
Block.sand
});

//Recipe for NOT Gate
ModLoader.AddRecipe(new ItemStack(ItemNOT, 6), new Object[] {
    "#XY", Character.valueOf('#'), Item.redstone, Character.valueOf('X'),
Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive
});

//Recipes for OR Gate
ModLoader.AddRecipe(new ItemStack(ItemOR, 4), new Object[] {
    "# ", "X#", "# ", Character.valueOf('#'), Item.redstone,
Character.valueOf('X'), Block.sand
});
ModLoader.AddRecipe(new ItemStack(ItemOR, 1), new Object[] {
    "#X", Character.valueOf('#'), ItemNOR, Character.valueOf('X'), ItemNOT
});

//Recipes for AND Gate
ModLoader.AddRecipe(new ItemStack(ItemAND, 4), new Object[] {
    "#Y ", " X#", "#Y ", Character.valueOf('#'), Item.redstone,
Character.valueOf('X'), Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive
});
ModLoader.AddRecipe(new ItemStack(ItemAND, 1), new Object[] {
    "#X", Character.valueOf('#'), ItemNAND, Character.valueOf('X'), ItemNOT
});

//Recipes for XOR Gate
ModLoader.AddRecipe(new ItemStack(ItemXOR, 1), new Object[] {
    "Y# ", " X", "Y# ", Character.valueOf('#'), ItemOR, Character.valueOf('X'),
ItemNAND, Character.valueOf('Y'), ItemNOT
});
ModLoader.AddRecipe(new ItemStack(ItemXOR, 1), new Object[] {
    "Y# ", " X", "Y# ", Character.valueOf('#'), ItemAND,
Character.valueOf('X'), ItemOR, Character.valueOf('Y'), ItemNOT
});
ModLoader.AddRecipe(new ItemStack(ItemXOR, 1), new Object[] {
    "#X", Character.valueOf('#'), ItemXNOR, Character.valueOf('X'), ItemNOT
});

//Recipe for NOR Gate
ModLoader.AddRecipe(new ItemStack(ItemNOR, 1), new Object[] {
    "#X", Character.valueOf('#'), ItemOR, Character.valueOf('X'), ItemNOT
});
ModLoader.AddRecipe(new ItemStack(ItemNOR, 4), new Object[] {
    "# ", "XY", "# ", Character.valueOf('#'), Item.redstone,
Character.valueOf('X'), Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive
});
ModLoader.AddRecipe(new ItemStack(ItemNOR, 1), new Object[] {
    "X ", " #", "X ", Character.valueOf('#'), ItemAND, Character.valueOf('X'),
ItemNOT
});

//Recipes for NAND Gate
ModLoader.AddRecipe(new ItemStack(ItemNAND, 1), new Object[] {
    "#X", Character.valueOf('#'), ItemAND, Character.valueOf('X'), ItemNOT
});
ModLoader.AddRecipe(new ItemStack(ItemNAND, 4), new Object[] {
    "#Y ", " XY", "#Y ", Character.valueOf('#'), Item.redstone,
Character.valueOf('X'), Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive
});
ModLoader.AddRecipe(new ItemStack(ItemNAND, 1), new Object[] {
```

```

        "X ", " #", "X ", Character.valueOf('#'), ItemOR, Character.valueOf('X'),
ItemNOT
    });

    //Recipes for XNOR Gate
    ModLoader.AddRecipe(new ItemStack(ItemXNOR, 1), new Object[] {
        "Y# ", " X", "Y# ", Character.valueOf('#'), ItemOR, Character.valueOf('X'),
ItemAND, Character.valueOf('Y'), ItemNOT
    });
    ModLoader.AddRecipe(new ItemStack(ItemXNOR, 1), new Object[] {
        "Y# ", " X", "Y# ", Character.valueOf('#'), ItemAND,
Character.valueOf('X'), ItemNOR, Character.valueOf('Y'), ItemNOT
    });
    ModLoader.AddRecipe(new ItemStack(ItemXNOR, 1), new Object[] {
        "#X", Character.valueOf('#'), ItemXOR, Character.valueOf('X'), ItemNOT
    });
}

// Tests for game version
public String Version()
{
    return "1.5_01";
}

//Block declaration
public static final Block BlockNOT;
public static final Block BlockOR;
public static final Block BlockAND;
public static final Block BlockXOR;
public static final Block BlockNOR;
public static final Block BlockNAND;
public static final Block BlockXNOR;
public static final Block BlockDiode;

//Render type declaration
int RenderId1;
public static int RenderGate = 200;

//Properties
private static Properties properties = new Properties();
public static boolean PropNewID;

public static int BlockNOTID;
public static int BlockORID;
public static int BlockANDID;
public static int BlockXORID;
public static int BlockNORID;
public static int BlockNANDID;
public static int BlockXNORID;
public static int BlockDiodeID;
public static int ItemNOTID;
public static int ItemORID;
public static int ItemANDID;
public static int ItemXORID;
public static int ItemNORID;
public static int ItemNANDID;
public static int ItemXNORID;
public static int ItemDiodeID;

//Texture declaration
public static int TextureNOTOn;
public static int TextureNOTOff;
public static int TextureOROn;
public static int TextureOROff;

```

```

public static int TextureNOROn;
public static int TextureNOROff;
public static int TextureDiodeOn;
public static int TextureDiodeOff;

//Icon declaration
public static int IconNOT;
public static int IconOR;
public static int IconAND;
public static int IconXOR;
public static int IconNOR;
public static int IconNAND;
public static int IconXNOR;
public static int IconDiode;

//Item declaration
public static Item ItemNOT;
public static Item ItemOR;
public static Item ItemAND;
public static Item ItemXOR;
public static Item ItemNOR;
public static Item ItemNAND;
public static Item ItemXNOR;
public static Item ItemDiode;

//Block and Item data
static
{
    LoadProperties();
    TextureNOTOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureNOTOff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureOROn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureOROff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureNOROn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureNOROff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureDiodeOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureDiodeOff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    IconNOT = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconOR = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconAND = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconXOR = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconNOR = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconNAND = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconXNOR = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    IconDiode = ModLoader.getUniqueSpriteIndex("/gui/items.png");

    BlockNOT = new BlockGateNOT(BlockNOTID,
147).setHardness(0.0F).setBlockName("NOT");
    BlockOR = new BlockGateOR(BlockORID, 147).setHardness(0.0F).setBlockName("OR");
    BlockAND = new BlockGateAND(BlockANDID,
147).setHardness(0.0F).setBlockName("AND");
    BlockXOR = new BlockGateXOR(BlockXORID,
147).setHardness(0.0F).setBlockName("XOR");
    BlockNOR = new BlockGateNOR(BlockNORID,
147).setHardness(0.0F).setBlockName("NOR");
    BlockNAND = new BlockGateNAND(BlockNANDID,
147).setHardness(0.0F).setBlockName("NAND");
    BlockXNOR = new BlockGateXNOR(BlockXNORID,
147).setHardness(0.0F).setBlockName("XNOR");
    BlockDiode = new BlockGateDiode(BlockDiodeID,
147).setHardness(0.0F).setBlockName("LGDiode");

    ItemNOT = (new ItemReed(ItemNOTID,
BlockNOT)).setIconIndex(IconNOT).setItemName("NOT");

```

mod_LogicalGates.java

```
ItemOR = (new ItemReed(ItemORID,
BlockOR)).setIconIndex(IconOR).setItemName("OR");
ItemAND = (new ItemReed(ItemANDID,
BlockAND)).setIconIndex(IconAND).setItemName("AND");
ItemXOR = (new ItemReed(ItemXORID,
BlockXOR)).setIconIndex(IconXOR).setItemName("XOR");
ItemNOR = (new ItemReed(ItemNORID,
BlockNOR)).setIconIndex(IconNOR).setItemName("NOR");
ItemNAND = (new ItemReed(ItemNANDID,
BlockNAND)).setIconIndex(IconNAND).setItemName("NAND");
ItemXNOR = (new ItemReed(ItemXNORID,
BlockXNOR)).setIconIndex(IconXNOR).setItemName("XNOR");
ItemDiode = (new ItemReed(ItemDiodeID,
BlockDiode)).setIconIndex(IconDiode).setItemName("LGDiode");
}

//Settings for handling of the property file
public static void LoadProperties()
{
    InputStream inputstream =
(mod_LogicalGates.class).getClassLoader().getResourceAsStream("MoareAI/LogicalGates.pro
perties");
    if(inputstream != null)
    {
        try
        {
            properties.load(inputstream);
            System.out.println("Succesfully loaded Proteties for MoareAI's Logical
Gates");

            PropNewID = Boolean.parseBoolean(properties.getProperty("NewIDs"));
            System.out.println((new StringBuilder()).append("Change IDs:
").append(properties.getProperty("ChangeIDs")).toString());
            if (PropNewID)
            {
                BlockNOTID =
Integer.parseInt(properties.getProperty("BlockNOTGate"));
                BlockORID =
Integer.parseInt(properties.getProperty("BlockORGate"));
                BlockANDID =
Integer.parseInt(properties.getProperty("BlockANDGate"));
                BlockXORID =
Integer.parseInt(properties.getProperty("BlockXORGate"));
                BlockNORID =
Integer.parseInt(properties.getProperty("BlockNORGate"));
                BlockNANDID =
Integer.parseInt(properties.getProperty("BlockNANDGate"));
                BlockXNORID =
Integer.parseInt(properties.getProperty("BlockXNORGate"));
                BlockDiodeID =
Integer.parseInt(properties.getProperty("BlockDiode"));
                ItemNOTID =
Integer.parseInt(properties.getProperty("ItemNOTGate"));
                ItemORID = Integer.parseInt(properties.getProperty("ItemORGate"));
                ItemANDID =
Integer.parseInt(properties.getProperty("ItemANDGate"));
                ItemXORID =
Integer.parseInt(properties.getProperty("ItemXORGate"));
                ItemNORID =
Integer.parseInt(properties.getProperty("ItemNORGate"));
                ItemNANDID =
Integer.parseInt(properties.getProperty("ItemNANDGate"));
                ItemXNORID =
Integer.parseInt(properties.getProperty("ItemXNORGate"));
                ItemDiodeID =
```

```

Integer.parseInt(properties.getProperty("ItemDiode"));
    }
    else
    {
        System.out.println("Old ID standard used");
        BlockNOTID = 225;
        BlockDiodeID = 226;
        BlockORID = 227;
        BlockANDID = 229;
        BlockNORID = 231;
        BlockNANDID = 233;
        BlockXORID = 235;
        BlockXNORID = 237;
        ItemDiodeID = 224;
        ItemNOTID = 225;
        ItemORID = 226;
        ItemANDID = 227;
        ItemNORID = 228;
        ItemNANDID = 229;
        ItemXORID = 230;
        ItemXNORID = 231;
    }
    System.out.println((new StringBuilder()).append("Block: NOT Gate ID:
").append(BlockNOTID).toString());
    System.out.println((new StringBuilder()).append("Block: OR Gate ID:
").append(BlockORID).toString());
    System.out.println((new StringBuilder()).append("Block: AND Gate ID:
").append(BlockANDID).toString());
    System.out.println((new StringBuilder()).append("Block: XOR Gate ID:
").append(BlockXORID).toString());
    System.out.println((new StringBuilder()).append("Block: NOR Gate ID:
").append(BlockNORID).toString());
    System.out.println((new StringBuilder()).append("Block: NAND Gate ID:
").append(BlockNANDID).toString());
    System.out.println((new StringBuilder()).append("Block: XNOR Gate ID:
").append(BlockXNORID).toString());
    System.out.println((new StringBuilder()).append("Block: Diode ID:
").append(BlockDiodeID).toString());
    System.out.println((new StringBuilder()).append("Item: NOT Gate ID:
").append(ItemNOTID).toString());
    System.out.println((new StringBuilder()).append("Item: OR Gate ID:
").append(ItemORID).toString());
    System.out.println((new StringBuilder()).append("Item: AND Gate ID:
").append(ItemANDID).toString());
    System.out.println((new StringBuilder()).append("Item: XOR Gate ID:
").append(ItemXORID).toString());
    System.out.println((new StringBuilder()).append("Item: NOR Gate ID:
").append(ItemNORID).toString());
    System.out.println((new StringBuilder()).append("Item: NAND Gate ID:
").append(ItemNANDID).toString());
    System.out.println((new StringBuilder()).append("Item: XNOR Gate ID:
").append(ItemXNORID).toString());
    System.out.println((new StringBuilder()).append("Item: Diode ID:
").append(ItemDiodeID).toString());
    return;
}
catch(IOException ioexception)
{
    System.out.println("Failed to load Properties for MoareAI's Logical
Gates");
}
}
}

```

```

//The rendering for the gates
public boolean RenderWorldBlock(RenderBlocks renderblocks, IBlockAccess
iblockaccess, int i, int j, int k, Block block, int l)
{
    if(l == RenderId1)
    {
        Tessellator tessellator = Tessellator.instance;
        int i2 = iblockaccess.getBlockMetadata(i, j, k);
        int i1 = BlockBed.getDirectionFromMetadata(i2);
        float f1 = 1.0F;
        float f4 = f1;
        float f5 = f1;
        float f6 = f1;
        tessellator.setColorOpaque_F(0,0,0);
        int j1 = block.getBlockTexture(iblockaccess, i, j, k, 0);
        int k1 = (j1 & 0xf) << 4;
        int l1 = j1 & 0xf0;
        double d = (float)k1 / 256F;
        double d1 = ((double)(k1 + 16) - 0.01D) / 256D;
        double d2 = (float)l1 / 256F;
        double d3 = ((double)(l1 + 16) - 0.01D) / 256D;
        double d40 = d;
        double d41 = d1;
        double d42 = d2;
        double d43 = d3;
        double d44 = d;
        double d45 = d1;
        double d46 = d2;
        double d47 = d3;
        if(i1 % 2 == 1)
        {
            d41 = d;
            d42 = d3;
            d44 = d1;
            d47 = d2;
            d43 = d3;
            d40 = d1;
            d45 = d;
            d46 = d2;
        } else
        if(i1 % 2 == 0)
        {
            d40 = d1;
            d43 = d3;
            d45 = d;
            d46 = d2;
        }
        double d4 = (double)i + block.minX;
        double d5 = (double)i + block.maxX;
        double d6 = (double)j + block.minY + 0.125D;
        double d7 = (double)k + block.minZ;
        double d8 = (double)k + block.maxZ;
        tessellator.addVertexWithUV(d4, d6, d8, d44, d46);
        tessellator.addVertexWithUV(d4, d6, d7, d40, d42);
        tessellator.addVertexWithUV(d5, d6, d7, d41, d43);
        tessellator.addVertexWithUV(d5, d6, d8, d45, d47);
        float f17 = block.getBlockBrightness(iblockaccess, i, j + 1, k);
        tessellator.setColorOpaque_F(f4 * f17, f5 * f17, f6 * f17);
        k1 = block.getBlockTexture(iblockaccess, i, j, k, 1);
        l1 = (k1 & 0xf) << 4;
        d = k1 & 0xf0;
        double d9 = (float)l1 / 256F;
        double d10 = ((double)(l1 + 16) - 0.01D) / 256D;
        double d11 = (float)d / 256F;
    }
}

```

```

double d12 = ((d + 16D) - 0.01D) / 256D;
double d13 = d9;
double d14 = d10;
double d15 = d11;
double d16 = d11;
double d17 = d9;
double d18 = d10;
double d19 = d12;
double d20 = d12;
if(i1 == 0)
{
    d14 = d9;
    d15 = d12;
    d17 = d10;
    d20 = d11;
} else
if(i1 == 2)
{
    d13 = d10;
    d16 = d12;
    d18 = d9;
    d19 = d11;
} else
if(i1 == 3)
{
    d13 = d10;
    d16 = d12;
    d18 = d9;
    d19 = d11;
    d14 = d9;
    d15 = d12;
    d17 = d10;
    d20 = d11;
}
double d21 = (double)i + block.minX;
double d22 = (double)i + block.maxX;
double d23 = (double)j + block.maxY;
double d24 = (double)k + block.minZ;
double d25 = (double)k + block.maxZ;
tessellator.addVertexWithUV(d22, d23, d25, d17, d19);
tessellator.addVertexWithUV(d22, d23, d24, d13, d15);
tessellator.addVertexWithUV(d21, d23, d24, d14, d16);
tessellator.addVertexWithUV(d21, d23, d25, d18, d20);
if(i2 % 2 == 1)
{
    int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 2) & 0xf) << 4;
    int k2 = block.getBlockTexture(iblockaccess, i, j, k, 2) & 0xf0;
    double d30 = (((double)j2 + block.minX * 16D) / 256D;
    double d31 = (((double)j2 + block.maxX * 16D) - 0.01D) / 256D;
    double d32 = (((double)k2 + block.minY * 16D) / 256D;
    double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
    double d34 = i + block.minX;
    double d35 = i + block.maxX;
    double d36 = j + block.minY;
    double d37 = j + block.maxY;
    double d38 = k + block.minZ;
    tessellator.addVertexWithUV(d34, d37, d38, d31, d32);
    tessellator.addVertexWithUV(d35, d37, d38, d30, d32);
    tessellator.addVertexWithUV(d35, d36, d38, d30, d33);
    tessellator.addVertexWithUV(d34, d36, d38, d31, d33);
}
if(i2 % 2 == 1)
{
    int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 3) & 0xf) << 4;

```



```

        int k2 = block.getBlockTexture(iblockaccess, i, j, k, 3) & 0xf0;
        double d30 = (((double)j2 + block.minX * 16D) / 256D);
        double d31 = (((double)j2 + block.maxX * 16D) - 0.01D) / 256D;
        double d32 = (((double)k2 + block.minY * 16D) / 256D);
        double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = i + block.minX;
        double d35 = i + block.maxX;
        double d36 = j + block.minY;
        double d37 = j + block.maxY;
        double d38 = k + block.maxZ;
        tessellator.addVertexWithUV(d34, d37, d38, d30, d32);
        tessellator.addVertexWithUV(d34, d36, d38, d30, d33);
        tessellator.addVertexWithUV(d35, d36, d38, d31, d33);
        tessellator.addVertexWithUV(d35, d37, d38, d31, d32);
    }
    if(i2 % 2 == 0)
    {
        int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 4) & 0xf) << 4;
        int k2 = block.getBlockTexture(iblockaccess, i, j, k, 4) & 0xf0;
        double d30 = (((double)j2 + block.minZ * 16D) / 256D);
        double d31 = (((double)j2 + block.maxZ * 16D) - 0.01D) / 256D;
        double d32 = (((double)k2 + block.minY * 16D) / 256D);
        double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = i + block.minX;
        double d35 = j + block.minY;
        double d36 = j + block.maxY;
        double d37 = k + block.minZ;
        double d38 = k + block.maxZ;
        tessellator.addVertexWithUV(d34, d36, d38, d31, d32);
        tessellator.addVertexWithUV(d34, d36, d37, d30, d32);
        tessellator.addVertexWithUV(d34, d35, d37, d30, d33);
        tessellator.addVertexWithUV(d34, d35, d38, d31, d33);
    }
    if(i2 % 2 == 0)
    {
        int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 5) & 0xf) << 4;
        int k2 = block.getBlockTexture(iblockaccess, i, j, k, 5) & 0xf0;
        double d30 = (((double)j2 + block.minZ * 16D) / 256D);
        double d31 = (((double)j2 + block.maxZ * 16D) - 0.01D) / 256D;
        double d32 = (((double)k2 + block.minY * 16D) / 256D);
        double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = i + block.maxX;
        double d35 = j + block.minY;
        double d36 = j + block.maxY;
        double d37 = k + block.minZ;
        double d38 = k + block.maxZ;
        tessellator.addVertexWithUV(d34, d35, d38, d30, d33);
        tessellator.addVertexWithUV(d34, d35, d37, d31, d33);
        tessellator.addVertexWithUV(d34, d36, d37, d31, d32);
        tessellator.addVertexWithUV(d34, d36, d38, d30, d32);
    }
    return true;
}
return false;
}
}

```