

```

package net.minecraft.src;

// Created by MoareAI

import java.util.Random;

public class BlockGateNOR extends Block
{
    //Class parameters
    protected BlockGateNOR(int id, int texture)
    {
        super(id, texture, Material.circuits);
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 0.125F, 1.0F);
        setLightValue(0.625F);
    }

    //Use render types
    public boolean renderAsNormalBlock()
    {
        return false;
    }

    //Block texture
    public int getBlockTextureFromSideAndMetadata(int side, int meta)
    {
        if(meta <= 3)
            return blockIndexInTexture = mod_LogicalGates.TextureNOROff;
        else
            return blockIndexInTexture = mod_LogicalGates.TextureNOROn;
    }

    //Block type (preset)
    public int getRenderType()
    {
        return mod_LogicalGates.RenderGate;
    }

    //Updates blockID
    public void updateTick(World world, int x, int y, int z, Random random)
    {
        int meta = world.getBlockMetadata(x, y, z);
        world.setBlockAndMetadataWithNotify(x, y, z, mod_LogicalGates.BlockNOR.blockID,
meta);
    }

    //The function. Sets metadata to +4 when the gate is to be turned on
    public void onNeighborBlockChange(World world, int x, int y, int z, int id)
    {
        if(!canBlockStay(world, x, y, z))
        {
            dropBlockAsItem(world, x, y, z, world.getBlockMetadata(x, y, z));
            world.setBlockWithNotify(x, y, z, 0);
            return;
        }
        int meta = world.getBlockMetadata(x, y, z);
        boolean InputA = func_InputA(world, x, y, z, meta);
        boolean InputB = func_InputB(world, x, y, z, meta);
        boolean InputC = func_InputC(world, x, y, z, meta);
        if(!(InputA || InputB || InputC))
        {
            if (meta <= 3)
                world.setBlockMetadataWithNotify(x, y, z, meta+4);
            } else
        {

```

# BlockGateNOR.java

```

        if (meta > 3)
            world.setBlockMetadataWithNotify(x, y, z, meta-4);
    }

    world.scheduleBlockUpdate(x, y, z, blockID, 0);
}

//Signal out
public boolean isPoweringTo(IBlockAccess iblockaccess, int x, int y, int z, int
side)
{
    int meta = iblockaccess.getBlockMetadata(x, y, z);
    if(meta == 4)
        return side == 3;
    if(meta == 5)
        return side == 4;
    if(meta == 6)
        return side == 2;
    if(meta == 7)
        return side == 5;
    else
        return false;
}

//Signal in
//Left
private boolean func_InputA(World world, int x, int y, int z, int meta)
{
    int side = meta%4;
    if (side == 0)
        return world.isBlockIndirectlyGettingPowered(x-1, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4);
    if (side == 1)
        return world.isBlockIndirectlyGettingPowered(x, y, z-1) ||
world.isBlockIndirectlyProvidingPowerTo(x, y, z-1, 2);
    if (side == 2)
        return world.isBlockIndirectlyGettingPowered(x+1, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y, z, 5);
    if (side == 3)
        return world.isBlockIndirectlyGettingPowered(x, y, z+1) ||
world.isBlockIndirectlyProvidingPowerTo(x, y, z+1, 3);
    return false;
}

//Back
private boolean func_InputB(World world, int x, int y, int z, int meta)
{
    int side = meta%4;
    if (side == 0)
        return world.isBlockIndirectlyGettingPowered(x, y, z+1) ||
world.isBlockIndirectlyProvidingPowerTo(x, y, z+1, 3);
    if (side == 1)
        return world.isBlockIndirectlyGettingPowered(x-1, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4);
    if (side == 2)
        return world.isBlockIndirectlyGettingPowered(x, y, z-1) ||
world.isBlockIndirectlyProvidingPowerTo(x, y, z-1, 2);
    if (side == 3)
        return world.isBlockIndirectlyGettingPowered(x+1, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y, z, 5);
    return false;
}

//Right

```

```

private boolean func_InputC(World world, int x, int y, int z, int meta)
{
    int side = meta%4;
    if (side == 0)
        return world.isBlockIndirectlyGettingPowered(x+1, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y, z, 5);
    if (side == 1)
        return world.isBlockIndirectlyGettingPowered(x, y, z+1) ||
world.isBlockIndirectlyProvidingPowerTo(x, y, z+1, 3);
    if (side == 2)
        return world.isBlockIndirectlyGettingPowered(x-1, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4);
    if (side == 3)
        return world.isBlockIndirectlyGettingPowered(x, y, z-1) ||
world.isBlockIndirectlyProvidingPowerTo(x, y, z-1, 2);
    return false;
}

//Rotation when placed
public void onBlockPlacedBy(World world, int x, int y, int z, EntityLiving
entityliving)
{
    int meta = ((MathHelper.floor_double(((double)((entityliving.rotationYaw * 4F) /
360F) + 0.5D) & 3) + 2) % 4;
    world.setBlockMetadataWithNotify(x, y, z, meta);
    int id = blockID;
    onNeighborBlockChange(world, x, y, z, id);
}

//Notifies neighbor blocks when added
public void onBlockAdded(World world, int x, int y, int z)
{
    world.notifyBlocksOfNeighborChange(x+1, y, z, blockID);
    world.notifyBlocksOfNeighborChange(x-1, y, z, blockID);
    world.notifyBlocksOfNeighborChange(x, y, z+1, blockID);
    world.notifyBlocksOfNeighborChange(x, y, z-1, blockID);
    world.notifyBlocksOfNeighborChange(x, y-1, z, blockID);
    world.notifyBlocksOfNeighborChange(x, y+1, z, blockID);
}

//Dropped
public int idDropped(int i, Random random)
{
    return mod_LogicalGates.ItemNOR.shiftedIndex;
}

//Tests if the block is solid
public boolean isOpaqueCube()
{
    return false;
}

//Tests if the block may provide power. Used for drawing Redstone Wires.
public boolean canProvidePower()
{
    return true;
}

//Tests where the block may be placed
public boolean canPlaceBlockAt(World world, int x, int y, int z)
{
    if(!world.isBlockOpaqueCube(x, y - 1, z))
        return false;
}

```

BlockGateNOR.java

```
        else
            return super.canPlaceBlockAt(world, x, y, z);
    }

    //Tests if the block may stay in the world
    public boolean canBlockStay(World world, int x, int y, int z)
    {
        if(!world.isBlockOpaqueCube(x, y - 1, z))
            return false;
        else
            return super.canBlockStay(world, x, y, z);
    }
}
```