

BlockLightbulb.java

```

package MoareAI.Class;

// Created by MoareAI

import java.util.Random;

public class BlockLightbulb extends BlockTorch
implements ITextureProvider
{
    //Class parameters
    public BlockLightbulb(int ID, int texture, float light)
    {
        super(ID, texture);
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 1.0F, 1.0F);
        setHardness(0.0F);
        setLightValue(light);
        disableStats();
        setTickOnLoad(true);
    }

    public String getTextureFile()
    {
        return mod_Lightbulb.BlockTexture;
    }

    //Turn on/off the lightbulb
    public void updateTick(World world, int x, int y, int z, Random random)
    {
        int meta = world.getBlockMetadata(x, y, z);
        boolean Input = func_Input(world, x, y, z, meta);
        if(Input)
        {
            world.setBlockAndMetadataWithNotify(x, y, z,
mod_Lightbulb.BlockLightbulbOn.blockID, meta);
        }
        else
        {
            world.setBlockAndMetadataWithNotify(x, y, z,
mod_Lightbulb.BlockLightbulbOff.blockID, meta);
        }
    }

    //Update block
    public void onNeighborBlockChange(World world, int x, int y, int z, int id)
    {
        super.onNeighborBlockChange(world, x, y, z, id);
        world.scheduleBlockUpdate(x, y, z, blockID, 0);
    }

    public void onBlockAdded(World world, int x, int y, int z)
    {
        if(world.getBlockMetadata(x, y, z) == 0)
        {
            super.onBlockAdded(world, x, y, z);
        }
        int id = blockID;
        onNeighborBlockChange(world, x, y, z, id);
    }

    //Signal in
    private boolean func_Input(World world, int x, int y, int z, int meta)
    {
        return (world.isBlockIndirectlyGettingPowered(x, y, z) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4) ||

```

BlockLightbulb.java

```
(world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4) ||
(world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z+1, 3) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z-1, 2)) && world.getBlockId(x-1, y,
z) == Block.redstoneWire.blockID) || (world.isBlockIndirectlyProvidingPowerTo(x, y,
z-1, 2) || ((world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z-1, 4) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 5)) && world.getBlockId(x, y,
z-1) == Block.redstoneWire.blockID) || (world.isBlockIndirectlyProvidingPowerTo(x+1,
y, z, 5) || ((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 3) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z+1, 2)) && world.getBlockId(x+1, y,
z) == Block.redstoneWire.blockID) || (world.isBlockIndirectlyProvidingPowerTo(x, y,
z+1, 3) || ((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z+1, 4) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z+1, 5)) && world.getBlockId(x, y,
z+1) == Block.redstoneWire.blockID));
    }

    //Item yielded upon drop
    public int idDropped(int i, Random random)
    {
        return mod_Lightbulb.ItemLightbulb.shiftedIndex;
    }

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

    //Deactivate Torch effects
    public void randomDisplayTick(World world, int i, int j, int k, Random random)
    {
        return;
    }
}
```