

```

package MoareAI.Class;

import net.minecraft.src.*;

//Created by MoareAI

public class RenderFunction
{
    public RenderFunction()
    {

    }

    public static void RenderFunctionModel(Tessellator tessellator, Block block,
    IBlockAccess iblockaccess, int x, int y, int z, float white, float light, int rotation)
    {
        tessellator.setColorOpaque_F(light * white, light * white, light * white);
        int j1 = block.getBlockTexture(iblockaccess, x, y, z, 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 d9 = d;
        double d10 = d1;
        double d11 = d2;
        double d12 = d2;
        double d13 = d;
        double d14 = d1;
        double d15 = d3;
        double d16 = d3;
        if(rotation == 1)
        {
            d10 = d;
            d11 = d3;
            d13 = d1;
            d16 = d2;
        } else
        if(rotation == 3)
        {
            d9 = d1;
            d12 = d3;
            d14 = d;
            d15 = d2;
        } else
        if(rotation == 2)
        {
            d9 = d1;
            d12 = d3;
            d14 = d;
            d15 = d2;
            d10 = d;
            d11 = d3;
            d13 = d1;
            d16 = d2;
        }
        double d4 = (double)x + block.minX;
        double d5 = (double)x + block.maxX;
        double d6 = (double)y + block.minY + 0.125D;
        double d7 = (double)z + block.minZ;
        double d8 = (double)z + block.maxZ;
        tessellator.addVertexWithUV(d4, d6, d8, d13, d15);
        tessellator.addVertexWithUV(d4, d6, d7, d9, d11);
    }
}

```

```

tessellator.addVertexWithUV(d5, d6, d7, d10, d12);
tessellator.addVertexWithUV(d5, d6, d8, d14, d16);

//Top
j1 = block.getBlockTexture(iblockaccess, x, y, z, 1);
k1 = (j1 & 0xf) << 4;
l1 = j1 & 0xf0;
d = (float)k1 / 256F;
d1 = ((double)(k1 + 16) - 0.01D) / 256D;
d2 = (float)l1 / 256F;
d3 = ((double)(l1 + 16D) - 0.01D) / 256D;
d9 = d;
d10 = d1;
d11 = d2;
d12 = d2;
d13 = d;
d14 = d1;
d15 = d3;
d16 = d3;
if(rotation == 3)
{
    d10 = d;
    d11 = d3;
    d13 = d1;
    d16 = d2;
} else
if(rotation == 1)
{
    d9 = d1;
    d12 = d3;
    d14 = d;
    d15 = d2;
} else
if(rotation == 0)
{
    d9 = d1;
    d12 = d3;
    d14 = d;
    d15 = d2;
    d10 = d;
    d11 = d3;
    d13 = d1;
    d16 = d2;
}
d4 = (double)x + block.minX;
d5 = (double)x + block.maxX;
d6 = (double)y + block.maxY;
d7 = (double)z + block.minZ;
d8 = (double)z + block.maxZ;
tessellator.addVertexWithUV(d5, d6, d8, d9, d11);
tessellator.addVertexWithUV(d5, d6, d7, d13, d15);
tessellator.addVertexWithUV(d4, d6, d7, d14, d16);
tessellator.addVertexWithUV(d4, d6, d8, d10, d12);

j1 = 4;

//Sides
if(rotation % 2 == 1)
{
    int j2 = (j1 & 0xf) << 4;
    int k3 = j1 & 0xf0;
    double d30 = ((double)j2 + block.minX * 16D) / 256D;
    double d31 = (((double)j2 + block.maxX * 16D) - 0.01D) / 256D;
    double d32 = ((double)k3 + block.minY * 16D) / 256D;

```

```

        double d33 = (((double)k3 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = x + block.minX;
        double d35 = x + block.maxX;
        double d36 = y + block.minY;
        double d37 = y + block.maxY;
        double d38 = z + 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);

        d34 = x + block.minX;
        d35 = x + block.maxX;
        d36 = y + block.minY;
        d37 = y + block.maxY;
        d38 = z + 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(rotation % 2 == 0)
    {
        int j2 = (j1 & 0xf) << 4;
        int k3 = j1 & 0xf0;
        double d30 = (((double)j2 + block.minZ * 16D) / 256D;
        double d31 = (((double)j2 + block.maxZ * 16D) - 0.01D) / 256D;
        double d32 = (((double)k3 + block.minY * 16D) / 256D;
        double d33 = (((double)k3 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = x + block.minX;
        double d35 = y + block.minY;
        double d36 = y + block.maxY;
        double d37 = z + block.minZ;
        double d38 = z + 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);

        d34 = x + block.maxX;
        d35 = y + block.minY;
        d36 = y + block.maxY;
        d37 = z + block.minZ;
        d38 = z + 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);
    }
}

//Render the input connection
public static void RenderInputConnection(Tessellator tessellator, Block block, int
x, int y, int z, float red, float light, int rotation)
{
    tessellator.setColorOpaque_F(red * light, 0.0F, 0.0F);
    int j1 = 1;
    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 + 16D) - 0.01D) / 256D;
    double d9 = d;

```

```

double d10 = d1;
double d11 = d2;
double d12 = d2;
double d13 = d;
double d14 = d1;
double d15 = d3;
double d16 = d3;
if(rotation == 3)
{
    d10 = d;
    d11 = d3;
    d13 = d1;
    d16 = d2;
} else
if(rotation == 1)
{
    d9 = d1;
    d12 = d3;
    d14 = d;
    d15 = d2;
} else
if(rotation == 0)
{
    d9 = d1;
    d12 = d3;
    d14 = d;
    d15 = d2;
    d10 = d;
    d11 = d3;
    d13 = d1;
    d16 = d2;
}
double d4 = (double)x + block.minX;
double d5 = (double)x + block.maxX;
double d6 = (double)y + block.maxY;
double d7 = (double)z + block.minZ;
double d8 = (double)z + block.maxZ;
tessellator.addVertexWithUV(d5, d6, d8, d9, d11);
tessellator.addVertexWithUV(d5, d6, d7, d13, d15);
tessellator.addVertexWithUV(d4, d6, d7, d14, d16);
tessellator.addVertexWithUV(d4, d6, d8, d10, d12);
}

//Render the inputs
public static void RenderConnectionPiece(Tessellator tessellator, Block block, int
x, int y, int z, float red, float light, int rotation, int direction)
{
    tessellator.setColorOpaque_F(red * light, 0.0F, 0.0F);
    int j1 = 5;
    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 + 16D) - 0.01D) / 256D;
    double d9 = 0;
    double d10 = 0;
    double d11 = 0;
    double d12 = 0;
    double d13 = 0;
    double d14 = 0;
    double d15 = 0;
    double d16 = 0;
    if(rotation == direction%4)

```

```

{
    d9 = d1;
    d12 = d3;
    d14 = d;
    d15 = d2;
    d10 = d;
    d11 = d3;
    d13 = d1;
    d16 = d2;
}
else
if(rotation == (direction+1)%4)
{
    d9 = d1;
    d10 = d1;
    d11 = d2;
    d12 = d3;
    d13 = d;
    d14 = d;
    d15 = d2;
    d16 = d3;
}
else
if(rotation == (direction+2)%4)
{
    d9 = d;
    d10 = d1;
    d11 = d2;
    d12 = d2;
    d13 = d;
    d14 = d1;
    d15 = d3;
    d16 = d3;
}
else
if(rotation == (direction+3)%4)
{
    d9 = d;
    d10 = d;
    d11 = d3;
    d12 = d2;
    d13 = d1;
    d14 = d1;
    d15 = d3;
    d16 = d2;
}
double d4 = (double)x + block.minX;
double d5 = (double)x + block.maxX;
double d6 = (double)y + block.maxY;
double d7 = (double)z + block.minZ;
double d8 = (double)z + block.maxZ;
tessellator.addVertexWithUV(d5, d6, d8, d9, d11);
tessellator.addVertexWithUV(d5, d6, d7, d13, d15);
tessellator.addVertexWithUV(d4, d6, d7, d14, d16);
tessellator.addVertexWithUV(d4, d6, d8, d10, d12);
}

//Render the inputs
public static void RenderInput(Tessellator tessellator, Block block, int x, int y,
int z, float red, float cyan, float light, int rotation, int direction)
{
    tessellator.setColorOpaque_F(red * light, cyan * light, cyan * light);
    int j1 = 2;
    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 + 16D) - 0.01D) / 256D;
double d9 = 0;
double d10 = 0;
double d11 = 0;
double d12 = 0;
double d13 = 0;
double d14 = 0;
double d15 = 0;
double d16 = 0;
if(rotation == direction%4)
{
    d9 = d1;
    d12 = d3;
    d14 = d;
    d15 = d2;
    d10 = d;
    d11 = d3;
    d13 = d1;
    d16 = d2;
}
else
if(rotation == (direction+1)%4)
{
    d9 = d1;
    d10 = d1;
    d11 = d2;
    d12 = d3;
    d13 = d;
    d14 = d;
    d15 = d2;
    d16 = d3;
}
else
if(rotation == (direction+2)%4)
{
    d9 = d;
    d10 = d1;
    d11 = d2;
    d12 = d2;
    d13 = d;
    d14 = d1;
    d15 = d3;
    d16 = d3;
}
else
if(rotation == (direction+3)%4)
{
    d9 = d;
    d10 = d;
    d11 = d3;
    d12 = d2;
    d13 = d1;
    d14 = d1;
    d15 = d3;
    d16 = d2;
}
double d4 = (double)x + block.minX;
double d5 = (double)x + block.maxX;
double d6 = (double)y + block.maxY;
double d7 = (double)z + block.minZ;

```

RenderFunction.java

```

    double d8 = (double)z + block.maxZ;
    tessellator.addVertexWithUV(d5, d6, d8, d9, d11);
    tessellator.addVertexWithUV(d5, d6, d7, d13, d15);
    tessellator.addVertexWithUV(d4, d6, d7, d14, d16);
    tessellator.addVertexWithUV(d4, d6, d8, d10, d12);
}

//Render the output
public static void RenderOutput (Tessellator tessellator, Block block, int x, int
y, int z, float red, float light, int rotation, int direction)
{
    float illuminate = light;
    if (red == 1.0F)
        illuminate = 1.0F;
    tessellator.setColorOpaque_F(red * illuminate, 0.0F, 0.0F);
    int j1 = 3;
    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 + 16D) - 0.01D) / 256D;
    double d9 = 0;
    double d10 = 0;
    double d11 = 0;
    double d12 = 0;
    double d13 = 0;
    double d14 = 0;
    double d15 = 0;
    double d16 = 0;
    if(rotation == direction%4)
    {
        d9 = d1;
        d12 = d3;
        d14 = d;
        d15 = d2;
        d10 = d;
        d11 = d3;
        d13 = d1;
        d16 = d2;
    }
    else
    if(rotation == (direction+1)%4)
    {
        d9 = d1;
        d10 = d1;
        d11 = d2;
        d12 = d3;
        d13 = d;
        d14 = d;
        d15 = d2;
        d16 = d3;
    }
    else
    if(rotation == (direction+2)%4)
    {
        d9 = d;
        d10 = d1;
        d11 = d2;
        d12 = d2;
        d13 = d;
        d14 = d1;
        d15 = d3;
        d16 = d3;
    }
}

```

```

    }
    else
    if(rotation == (direction+3)%4)
    {
        d9 = d;
        d10 = d;
        d11 = d3;
        d12 = d2;
        d13 = d1;
        d14 = d1;
        d15 = d3;
        d16 = d2;
    }
    double d4 = (double)x + block.minX;
    double d5 = (double)x + block.maxX;
    double d6 = (double)y + block.maxY;
    double d7 = (double)z + block.minZ;
    double d8 = (double)z + block.maxZ;
    tessellator.addVertexWithUV(d5, d6, d8, d9, d11);
    tessellator.addVertexWithUV(d5, d6, d7, d13, d15);
    tessellator.addVertexWithUV(d4, d6, d7, d14, d16);
    tessellator.addVertexWithUV(d4, d6, d8, d10, d12);
}
}

```