**SimElectricity（模拟电子）mod企划**

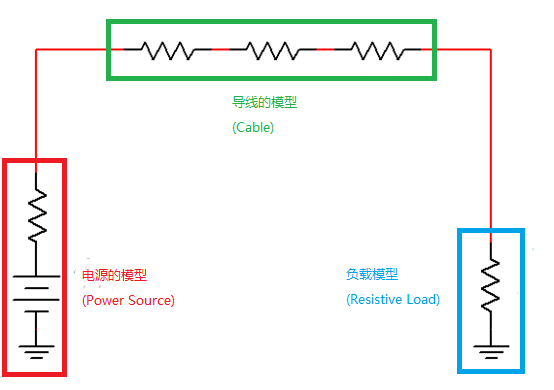
By 串亖の六花 and Aurora(小光)

本人也不是专业的Coder 因此可能有些观点上的错误 请大家指出

这次是实现一个能源网的仿真 需要能仿真电源 导线（有电阻）和负载（纯电阻）这三者的串并联

单位全部采用SI单位制(国际单位制) 电压-伏特,电流-安培,电功率-瓦特,电功-焦耳

三大器件



* 电源

这里的电源模型为1个理想电源和1个电阻的组合,电源发热过多会爆炸

* 导线

导线有电阻和绝缘强度:

1.过大的电流会导致电线熔毁

2.过高的电压（对地）会击穿绝缘对周围生物造成伤害

* 负载

所有负载均考虑为纯电阻性负载 负载有最大允许电压 超过可爆炸或者损坏 负载也可以有最低工作电压 负载做的攻用P=(U^2)/R计算

电源和负载一端接地,导线任何地方都不接地,所有地的节点都素同一个

能源网依靠节点网络进行计算工作

所有对能源网的仿真均在服务器端进行

**三大事件**

* TileAttachEvent

当一个用电器要加入能源网时需要发出此事件 比如被放置或者开关方块的闭合

这个事件会重新生成节点网络

参数是那个机器的TileEntity

三大器件都需要在TileEntity里的updateEntity和invalidate里发出这两个事件 详见SimElectricity.Blocks. TileQuantumGenerator.java:

protected boolean isAddedToEnergyNet = false;

@Override

public void updateEntity() {

if (!worldObj.isRemote &&!isAddedToEnergyNet) { //必须在服务端完成这些操作

Util.postTileAttachEvent(this); //发送加入能源网请求

this.isAddedToEnergyNet=true;

………

}

}

* TileDetachEvent

当一个用电器要离开能源网时需要发出此事件比如被拆掉被炸毁或者开关方块的断开

这个事件会重新生成节点网络

参数是那个机器的TileEntity

三大器件都需要在TileEntity里的updateEntity和invalidate里发出这两个事件 详见SimElectricity.Blocks. TileQuantumGenerator.java:

@Override

public void invalidate() {

if (!worldObj.isRemote &isAddedToEnergyNet) //必须在服务端完成这些操作

Util.postTileDetachEvent(this);

}

* TileChangeEvent

当一个电器的参数发生变化的时候需要发出这个事件

比如电池电量耗尽 机器开始/停止工作

这个事件不重新生成节点网络 只是更新部分信息 比如能否发出能量和电阻值等

这样能减少不必要的运算量

在simElectricity.Blocks.TileQuantumGenerator.java中, 当服务端收到客户端的请求之后, 电源参数已经变成,此时需要发送本事件才能通知能源网,以确保仿真的正确进行:

@Override

public void onClient2ServerUpdate(String field, Object value, short type) {

if(field.contains("outputVoltage"))

Util.postTileChangeEvent(this); //通知能源网这个节点发生参数变更

}

**三大接口**

* IBaseComponent

这个素一切的基石 只要素IBaseComponent就可以算本mod的实体 剩下两个接口都继承于这个

|  |  |
| --- | --- |
| float getResistance(); | 用来得到负载或者导线的电阻 对于电源则是内阻 |
| int getMaxPowerDissipation(); | 用来得到负载最大的允许消耗 |
| void onOverloaded(); | 发生在该元件消耗过大时 即Voltage^2/ getResistance()>getMaxPowerDissipation();时  用来处理电源过载 电源过流 负载超功率工作的情况 |

* IConductor

导线需要继承这个接口

|  |  |
| --- | --- |
| int getInsulationBreakdownVoltage() | 用于返回导线绝缘层的最大允许电压 0对应无限 |
| void onInsulationBreakdown(); | 当绝缘层被击穿时发生(getInsulationBreakdownVoltage()>0&&voltage> getInsulationBreakdownVoltage()) 实现电击生物或者起火等 |

* IEnergyTile

电源和负载需要继承这个接口(区别只在getOutputVoltage,负载需要直接返回0)

|  |  |
| --- | --- |
| ForgeDirection getFunctionalSide(); | 返回机器用来交换能量的面,只能有一面连接电线 |
| float getOutputVoltage(); | 对于电源,返回内部理想电压源的电压,对于负载,直接返回0 |
| float getMaxSafeVoltage () | 用于返回电器的最大输入电压 0对应无限 |
| void onOverVoltage (); | 当机器输入超压时触发(getMaxSafeVoltage ()>0&&voltage> getMaxSafeVoltage ()) 实现机器爆炸等 |

**EnergyNet详细工作原理阐述**

EnergyNet.java实现大部分的仿真工作,仿真工作限在服务端进行

* 内有1个Flag：calc 当收到TileAttachEvent 或者TileDetachEvent时，flag被=true，在下个tick中更新节点网络
* 当收到TileChangeEvent时 仅仅更新数据
* 仿真靠JGraph生成各实体之间的连接关系然后通过矩阵计算得到所有的节点电压实现
* 如果仿真网络中元件和参数没任何变化（没收到任何三大事件） 那么各负载的输入输出功率不变 每tick中的功耗能通过Voltage^2/ getResistance()计算（在机器的TileEntity中实现）
* 游戏中所有包含IBaseComponent的方块被抽象为节点，节点与相邻节点间的电阻取两节点电阻之和
* 基于游戏地图实现方式的限定，每个节点最多可拥有6（仅垂直相邻）个相邻节点
* 优化过程中相邻节点数量为2的节点被视为假节点，节点电阻与相邻电阻相加后跳过运算(待实现)
* 优化过程中相邻节点数量小于2的节点被视为死节点，与之相邻的所有假节点同样被视为死节点，死节点跳过运算(待实现)
* 每个电源和负载只能有一面与能源网互动,在getFunctionalSide中设置
* 不同电压的电源并联可能会产生倒灌并损坏

问题：变压器和储电器这种又接收能量又发射能量的东西如何考虑？

为此,窝设想了一种IComplexEnergyTile的东西,在一个坐标内存在两个电路元件，甚至可以连接在两个独立电网之间。IComplexEnergyTile允许有一个输入和一个输出。

* IComplexEnergyTile基于IEnergyTile:

|  |  |
| --- | --- |
| float getResistance(); | 用来得到输入等效电阻 |
| int getMaxPowerDissipation(); | 用来得到最大允许的功率损耗,即输出电阻上消耗的能量 |
| void onOverloaded(); | 发生在该元件消耗过大时 即(InternalVoltage-Voltage)^2/ getResistance()>getMaxPowerDissipation();时  用来处理输出过载 输出过流的情况 |
| ForgeDirection getFunctionalSide(); | 返回机器用来输入能量的面 |
| float getOutputVoltage(); | 返回内部理想电压源的输出电压 |
| float getMaxSafeVoltage () | 用于返回电器的最大输入电压 0对应无限 |
| void onOverVoltage (); | 当机器输入超压时触发(getMaxSafeVoltage ()>0&&voltage> getMaxSafeVoltage ()) 实现机器爆炸等 |
| ForgeDirection getOutputSide(); | 返回机器用来输出能量的面 |
| float getOutputResistance(); | 返回机器的输出等效电阻 |
| Float getMaxOutputSafeVoltage() | 返回输出侧的最大允许电压 |

由于目前能源网中电路元件和坐标一一对应 暂时无法实现IComplexEnergyTile

**网络同步部分**

由于素网络mod，因此需要频繁的进行客户端和服务端的同步操作，目前同步操作只限TileEntity中进行，PacketPipeline用来完成网络包的发送与管理，PacketTileEntityFieldUpdate用来完成处理和接受TileEntity的同步包。

在simElecticity.API.Util.java中, updateTileEntityField (TileEntity,String)用来完成从服务器更新客户端的指定的TileEntity的指定属性的同步, updateTileEntityFieldToServer(TileEntity,String) 用来完成从客户端更新服务器的指定的TileEntity的指定属性的同步, updateTileEntityField比updateTileEntityFieldToServer更常用一些,目前updateTileEntityFieldToServer只在QuantumGenerator的GUI里使用过。

需要注意的是, updateTileEntityFieldToServer和updateTileEntityField只能同步如下这些类型的属性(未来可能会增加,未知类型会报错):

Boolean,Integer,float,ForgeDirection,Boolean[6]

updateTileEntityField和updateTileEntityFieldToServer可以用来同步任何TileEntity的属性，包括非本mod的TileEntity

实体可以实现(implements) simElectricity.API.ISyncPacketHandler来实现收到并且应用属性的同步更新后的一些后续操作, onServer2ClientUpdate为客户端收到服务端更新之后的操作(在客户端处理), onClient2ServerUpdate为服务端收到客户端更新之后的操作(在服务端处理)。

@Override

public void onServer2ClientUpdate(String field, Object value, short type) {

if(field.contains("functionalSide")) //使用contain(“”)而且不是==,否则会不正常

worldObj.markBlockForUpdate(xCoord, yCoord, zCoord); //在客户端通知方块重新渲染(因为修改了面朝的方向)

}

@Override

public void onClient2ServerUpdate(String field, Object value, short type) {

if(field.contains("outputVoltage")) //使用contain(“”)而且不是==,否则会不正常

Util.postTileChangeEvent(this); //在服务端通知能源网参数变更(outputVoltage)

}

现在网络同步部分已经非常稳定并且可以自动进行,无需干预其内部操作

**后话**

simElectricity.API.Util.java中可以找到很多有用的方法

目前 如果一个导线没和任何电源或者负载连接 则会导致整个能源网停止工作 原因未知