package application.File;

import java.util.ArrayList;

import application.Model.\*;

import javafx.animation.Animation;

import javafx.animation.KeyFrame;

import javafx.animation.ParallelTransition;

import javafx.animation.PathTransition;

import javafx.animation.SequentialTransition;

import javafx.animation.Timeline;

import javafx.animation.TranslateTransition;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.scene.Group;

import javafx.scene.Scene;

import javafx.scene.paint.Color;

import javafx.scene.paint.CycleMethod;

import javafx.scene.paint.RadialGradient;

import javafx.scene.paint.Stop;

import javafx.scene.shape.Circle;

import javafx.scene.shape.CubicCurveTo;

import javafx.scene.shape.MoveTo;

import javafx.scene.shape.Path;

import javafx.stage.Stage;

import javafx.util.Duration;

public class RandomlyGeneratedShow extends Application {

Stage stage=new Stage();

Scene scene=new Scene(new Group(),450,350);

Group root =(Group) scene.getRoot();

private int NucleusCount; //原子核个数

private NucleusModel []Nucleus;//原子核模型

private NeutronModel Neutron;//中子模型

private ArrayList list =new ArrayList();

private ArrayList Neutronlist =new ArrayList();

@Override

public void start(Stage primaryStage) {

int i,j;

primaryStage.setTitle("随机生成的连锁反应动画");

try{

//初始化中子

Neutron=new NeutronModel();

Neutron.setno(1);

Neutronlist.add(Neutron);

//随机生成原子核数目

NucleusCount=(int) ((Math.random()\*20))+10;

//System.out.println("NucleusCount:"+NucleusCount);

Nucleus=new NucleusModel[NucleusCount];

//初始化原子核

for(i=0;i<NucleusCount;i++) {

Nucleus[i] =new NucleusModel();

Nucleus[i].setNo(i);

}

//将原子核加入面板

for(i=0;i<NucleusCount;i++) {

root.getChildren().add(Nucleus[i].getCircle());

list.add(Nucleus[i]);

}

//设置时间轴

Timeline time=new Timeline();

//初始化关键帧

time=new Timeline();

time.setCycleCount(Animation.INDEFINITE);//indefinite:不确定的。

//设置关键帧

KeyFrame moveNeurtion = new KeyFrame(Duration.seconds(.01500),

new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent event) {

//碰到边界改变方向

changevelocity(Neutronlist);

//判断是否发生碰撞

isCollision(list,Neutronlist);

}

});

time.getKeyFrames().add(moveNeurtion);

time.play();

root.getChildren().add(Neutron.getCircle());

primaryStage.setScene(scene);

primaryStage.show();

}catch(Exception e) {

e.printStackTrace();

}

}

public void isCollision(ArrayList list,ArrayList Neutronlist) {

int i,no=0,j;

int NeutronX,NeutronY;

NucleusModel nucleus =new NucleusModel();

NeutronModel Neutron= new NeutronModel();

for(i=0;i<Neutronlist.size();i++) {

for(j=0;j<list.size();j++){

Neutron=(NeutronModel)Neutronlist.get(i);

nucleus=(NucleusModel)list.get(j);

NeutronX=Neutron.getX();

NeutronY=Neutron.getY();

if((NeutronX-nucleus.getX())\*(NeutronX-nucleus.getX())

+(NeutronY-nucleus.getY())\*(NeutronY-nucleus.getY())

<=30\*30) {

no=nucleus.getNo();

Nucleus[no].setColor();

productNeutron(no);

list.remove(j);

NeutronModel neutron=new NeutronModel();

neutron.setXCoordinates(NeutronX);

neutron.setYCoordinates(NeutronY);

Neutronlist.add(neutron);

root.getChildren().add(neutron.getCircle());

}

}

}

}

//通过函数获得控件的边界位置，以此判断控件的坐标，是否需要换速度方向

public void changevelocity(ArrayList Neutronlist) {

int i;

for(i=0;i<Neutronlist.size();i++) {

NeutronModel neutron=(NeutronModel)Neutronlist.get(i);

Neutronlist.remove(i);

Circle circle=neutron.getCircle();

double xMin = circle.getBoundsInParent().getMinX();

double yMin = circle.getBoundsInParent().getMinY();

double xMax = circle.getBoundsInParent().getMaxX();

double yMax = circle.getBoundsInParent().getMaxY();

if(xMin<0 || xMax>450){

neutron.setXvelocity(neutron.getXVelocity()\*-1);

}

if (yMin < 0 || yMax > 350) {

neutron.setYvelocity(neutron.getYVelocity()\*-1);

}

//改变了中子的坐标后将新的中子返回给模型

circle.setTranslateX(circle.getTranslateX() + neutron.getXVelocity());

circle.setTranslateY(circle.getTranslateY() + neutron.getYVelocity());

neutron.setCircle(circle);

Neutronlist.add(i, neutron);

}

}

//碰撞后产生两个小的原子核，两个原子核直线移动一段距离

public void productNeutron(int no) {

int X,Y,Xv,Yv;

Path path=new Path();

X=Nucleus[no].getX();

Y=Nucleus[no].getY();

Xv=Neutron.getXVelocity();

Yv=Neutron.getYVelocity();

Nucleus nucleus1=new Nucleus(X,Y,Xv+10,Yv+10);

Nucleus nucleus2=new Nucleus(X,Y,Xv+10,Yv+10);

Circle circle1=nucleus1.getCircle();

Circle circle2=nucleus2.getCircle();

//第一个原子核的移动轨迹

TranslateTransition translateTransition1 =

new TranslateTransition(Duration.millis(200), circle1);

translateTransition1.setFromX(20);

translateTransition1.setToX(50);

translateTransition1.setToY(-50);

translateTransition1.setCycleCount(1);

translateTransition1.setAutoReverse(true);

//第二个原子核的移动轨迹

TranslateTransition translateTransition2 =

new TranslateTransition(Duration.millis(200), circle2);

translateTransition2.setFromX(20);

translateTransition2.setToX(50);

translateTransition2.setToY(50);

translateTransition2.setCycleCount(1);

translateTransition2.setAutoReverse(true);

//ParallelTransition同时运行多个animation

ParallelTransition parallelTransition=new ParallelTransition(translateTransition1,translateTransition2);

parallelTransition.setCycleCount(1);

parallelTransition.play();

root.getChildren().add(nucleus1.getCircle());

root.getChildren().add(nucleus2.getCircle());

}

public void showWindows() throws Exception{

start(stage);

}

}