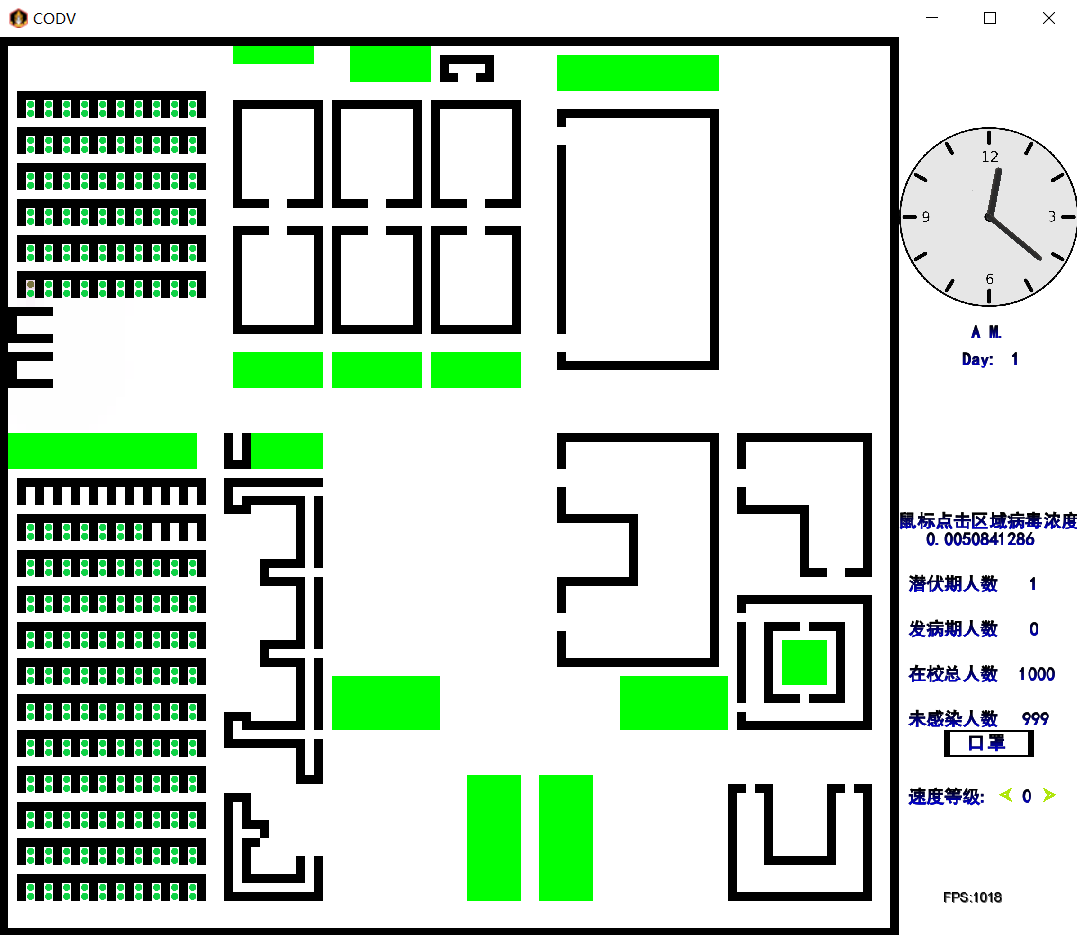
1. **感染机制**
2. 潜伏期：病毒潜伏时间为1-8天，潜伏期过后学生将会进 入发病期。
3. 发病期：发病后1-2天学生会被送往医院离开学校，同时发病期传染他人概率为潜伏期的5倍
4. 口罩：学生戴上口罩后，感染者散布病毒的速度为原来的0.2倍，健康者染病概率为原来的0.6倍
5. 图中的小点表示学生，其中绿色代表健康，褐色代表潜伏，红色代表发病

2图中的绿色块代表绿化，紫色块代表区域病毒浓度（病毒浓度越高，则颜色越深）

1. 初始条件为学校共有1000名学生，其中一名学生处于潜伏期，其余999名学生正常

初始情况截图：



1. 所有下列表格数据均采取每天00：00时数据

**二.具体实验**

**(一)未戴口罩：**

**1.第一次实验：**

**表格：**

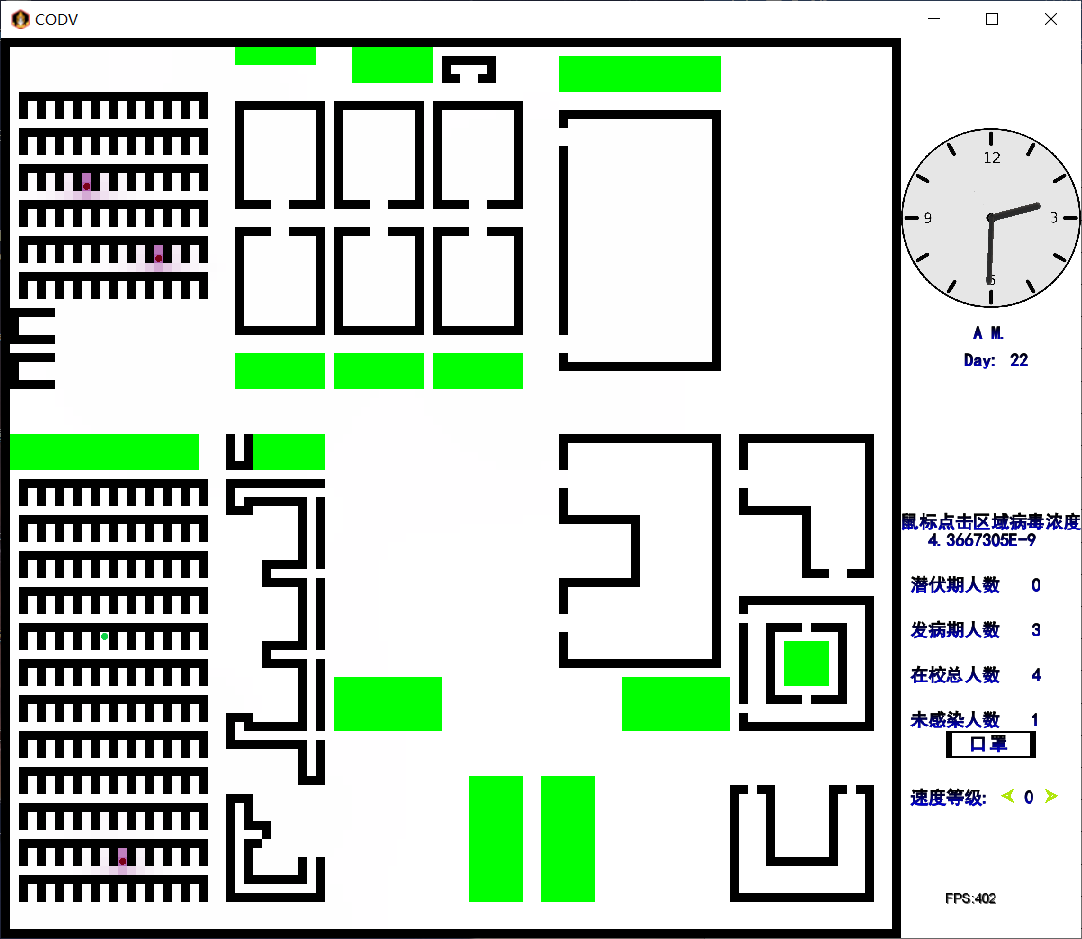
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 天数 | 潜伏期人数 | 发病期人数 | 在校总人数 | 未感染人数 |
| 1 | 1 | 0 | 1000 | 999 |
| 2 | 3 | 0 | 1000 | 997 |
| 3 | 5 | 0 | 1000 | 995 |
| 4 | 8 | 0 | 1000 | 992 |
| 5 | 15 | 1 | 1000 | 984 |
| 6 | 23 | 2 | 1000 | 975 |
| 7 | 57 | 11 | 999 | 931 |
| 8 | 158 | 9 | 992 | 825 |
| 9 | 317 | 15 | 987 | 655 |
| 10 | 510 | 45 | 980 | 425 |
| 11 | 643 | 86 | 952 | 223 |
| 12 | 691 | 130 | 904 | 83 |
| 13 | 619 | 172 | 822 | 31 |
| 14 | 514 | 196 | 718 | 8 |
| 15 | 380 | 204 | 586 | 2 |
| 16 | 257 | 193 | 452 | 2 |
| 17 | 159 | 158 | 318 | 1 |
| 18 | 79 | 128 | 208 | 1 |
| 19 | 32 | 88 | 121 | 1 |
| 20 | 6 | 44 | 51 | 1 |
| 21 | 2 | 19 | 22 | 1 |
| 22 | 1 | 3 | 5 | 1 |
| 23 | 0 | 1 | 2 | 1 |
| 24 | 0 | 0 | 1 | 1 |
| 25 | 0 | 0 | 1 | 1 |
| 26 | 0 | 0 | 1 | 1 |
| 27 | 0 | 0 | 1 | 1 |
| 28 | 0 | 0 | 1 | 1 |
| 29 | 0 | 0 | 1 | 1 |
| 30 | 0 | 0 | 1 | 1 |

**折线图：**

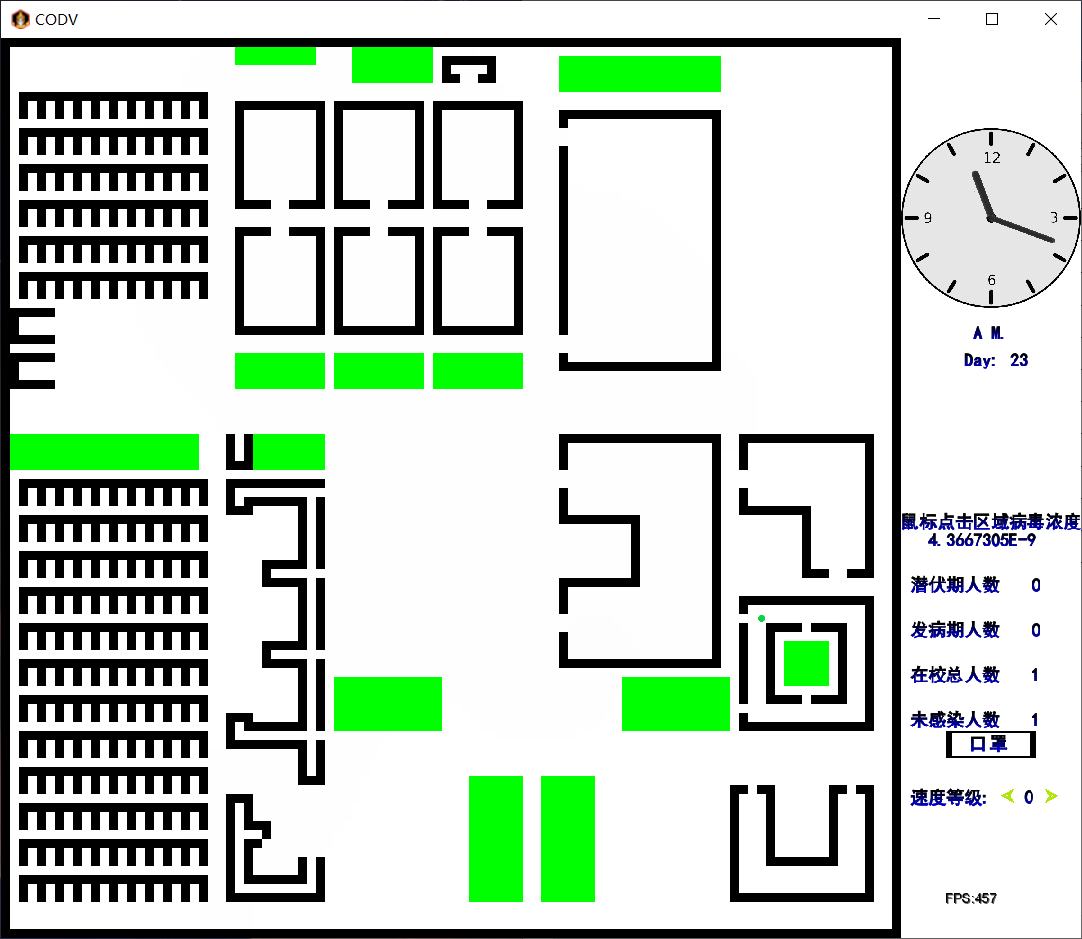
**柱状图：**

**.部分实验截图**

a.第22天02：30，潜伏期人数清零



b.第23天11：17，所有发病患者均送往医院，学校剩余1人健康



**2.第二次实验**

**表格：**

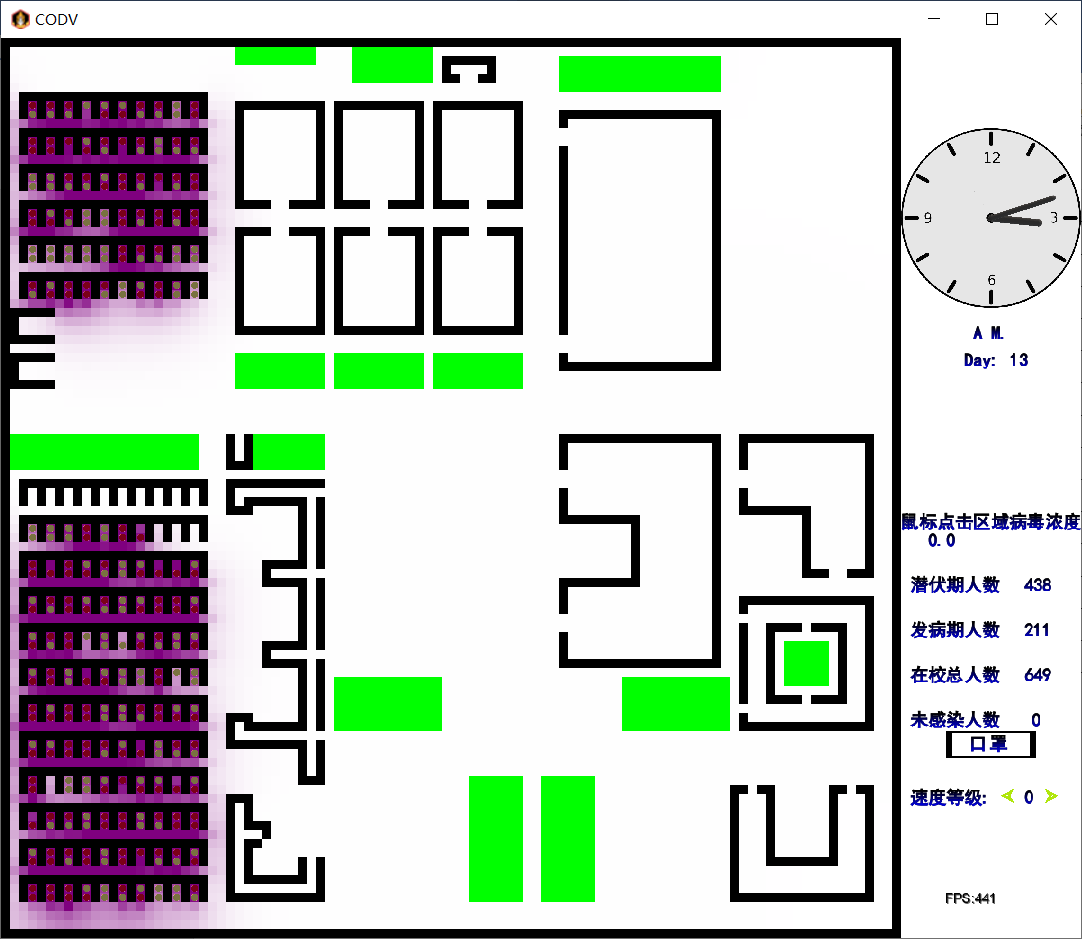
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 天数 | 潜伏期人数 | 发病期人数 | 在校总人数 | 未感染人数 |
| 1 | 1 | 0 | 1000 | 999 |
| 2 | 3 | 0 | 1000 | 997 |
| 3 | 5 | 0 | 1000 | 995 |
| 4 | 19 | 0 | 1000 | 981 |
| 5 | 44 | 1 | 1000 | 955 |
| 6 | 103 | 5 | 1000 | 892 |
| 7 | 224 | 17 | 998 | 757 |
| 8 | 395 | 29 | 989 | 565 |
| 9 | 604 | 63 | 969 | 302 |
| 10 | 723 | 90 | 928 | 115 |
| 11 | 689 | 143 | 878 | 46 |
| 12 | 585 | 188 | 785 | 12 |
| 13 | 452 | 205 | 658 | 1 |
| 14 | 316 | 209 | 525 | 0 |
| 15 | 199 | 188 | 387 | 0 |
| 16 | 99 | 151 | 250 | 0 |
| 17 | 36 | 114 | 150 | 0 |
| 18 | 11 | 57 | 68 | 0 |
| 19 | 3 | 20 | 23 | 0 |
| 20 | 1 | 4 | 5 | 0 |
| 21 | 0 | 4 | 4 | 0 |
| 22 | 0 | 1 | 1 | 0 |
| 23 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 |
| 28 | 0 | 0 | 0 | 0 |
| 29 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 |

**折线图：**

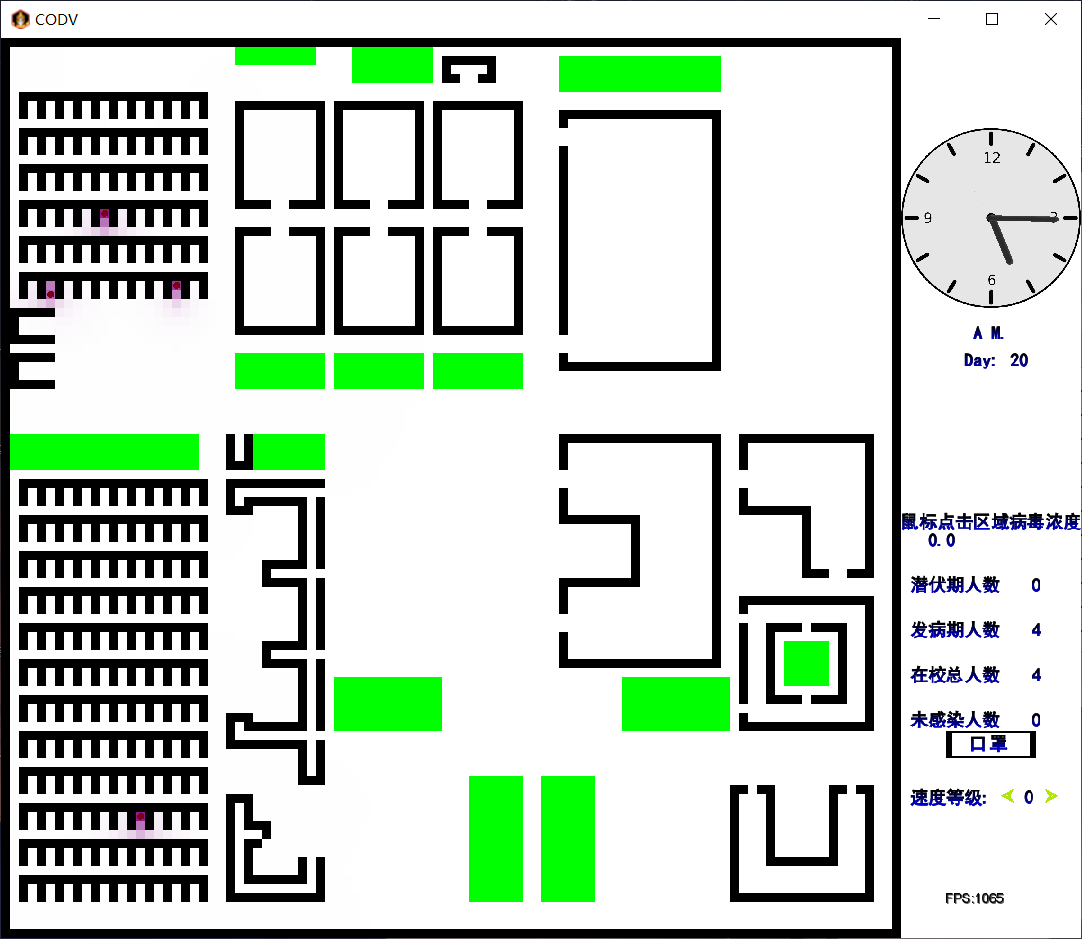
**柱状图：**

**部分实验截图**

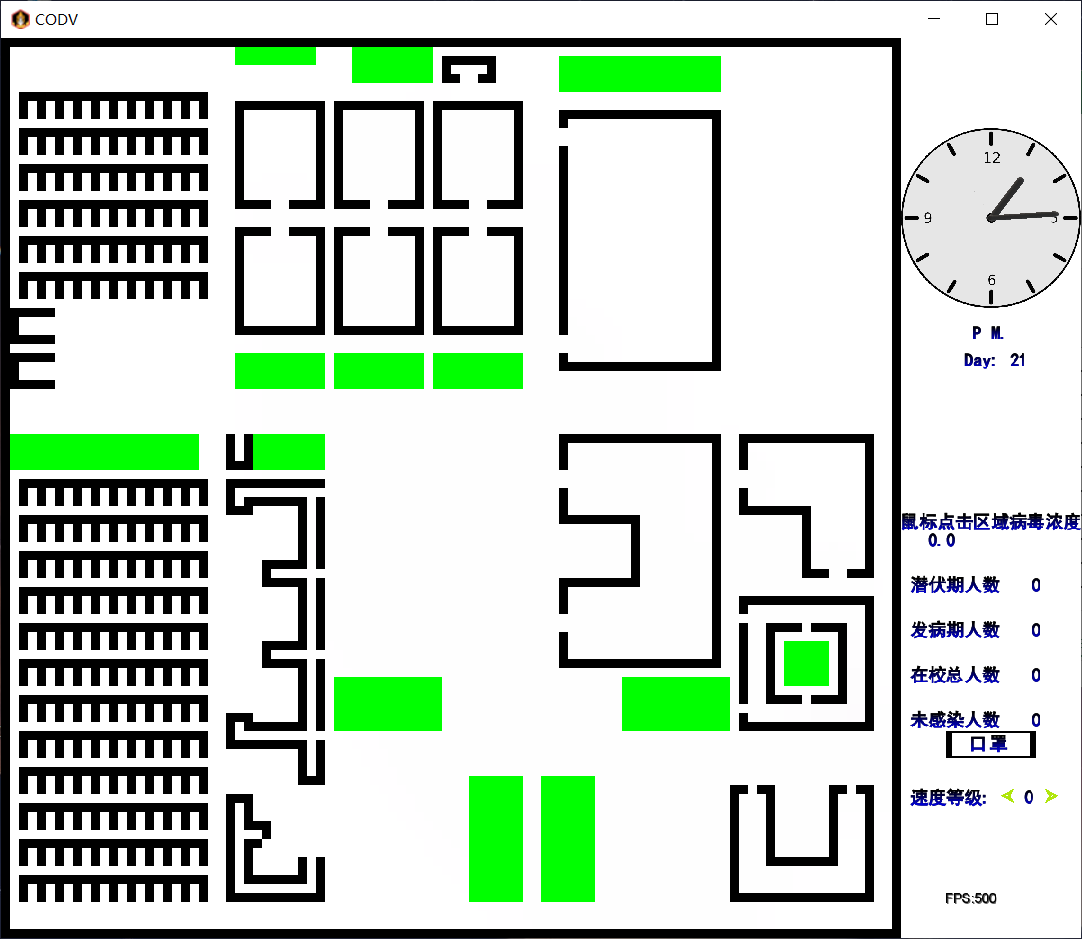
a.第13天03：12所有学生均被感染



b.第20日05：15，所有潜伏期学生均患病



c.第21日13：14所有患病学生均被送往医院



**(二)戴口罩：**

**1.第一次实验：**

**表格：**

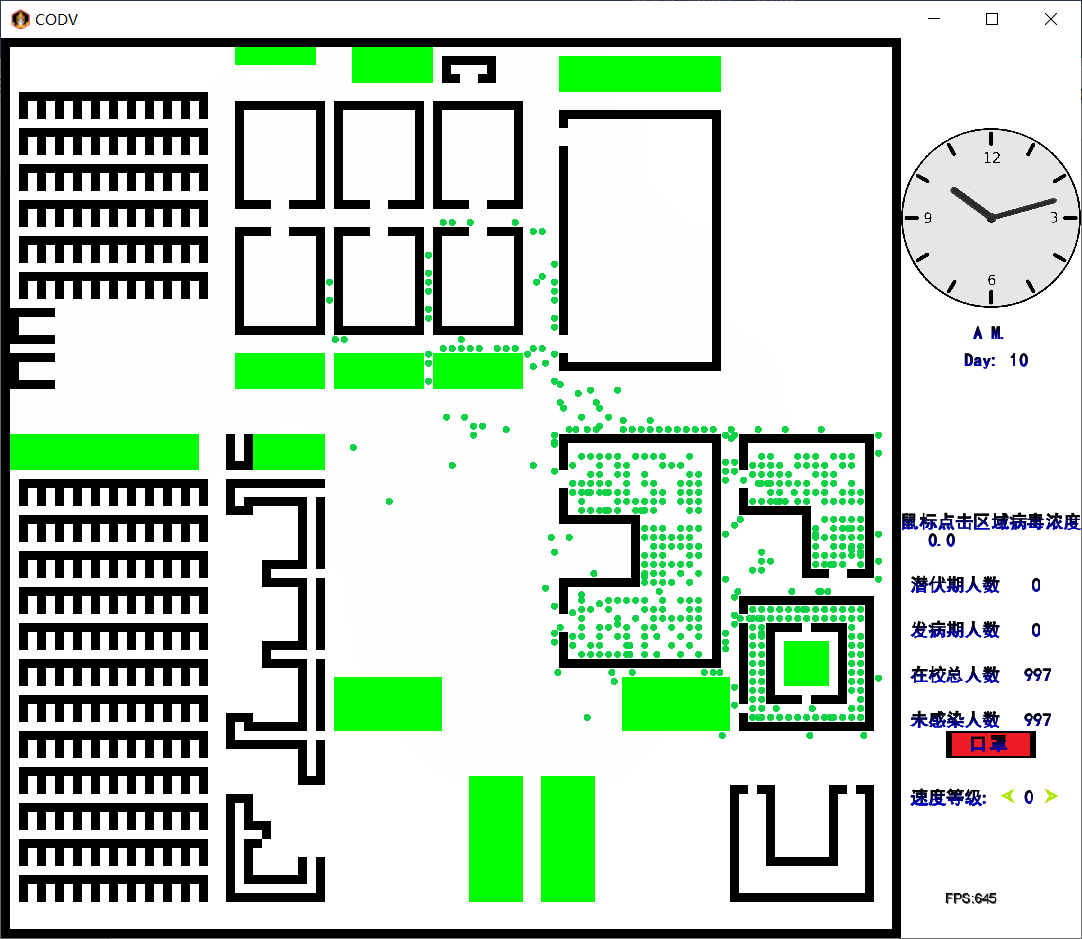
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 潜伏期人数 | 发病期人数 | 在校总人数 | 未感染人数 |
| 1 | 1 | 0 | 1000 | 999 |
| 2 | 1 | 0 | 1000 | 999 |
| 3 | 1 | 0 | 1000 | 999 |
| 4 | 0 | 1 | 1000 | 999 |
| 5 | 1 | 1 | 1000 | 998 |
| 6 | 2 | 0 | 999 | 997 |
| 7 | 1 | 1 | 999 | 997 |
| 8 | 1 | 0 | 998 | 997 |
| 9 | 0 | 1 | 998 | 997 |
| 10 | 0 | 1 | 998 | 997 |
| 11 | 0 | 0 | 998 | 997 |
| 12 | 0 | 0 | 998 | 997 |
| 13 | 0 | 0 | 998 | 997 |
| 14 | 0 | 0 | 998 | 997 |
| 15 | 0 | 0 | 998 | 997 |
| 16 | 0 | 0 | 998 | 997 |
| 17 | 0 | 0 | 998 | 997 |
| 18 | 0 | 0 | 998 | 997 |
| 19 | 0 | 0 | 998 | 997 |
| 20 | 0 | 0 | 998 | 997 |
| 21 | 0 | 0 | 998 | 997 |
| 22 | 0 | 0 | 998 | 997 |
| 23 | 0 | 0 | 998 | 997 |
| 24 | 0 | 0 | 998 | 997 |
| 25 | 0 | 0 | 998 | 997 |
| 26 | 0 | 0 | 998 | 997 |
| 27 | 0 | 0 | 998 | 997 |
| 28 | 0 | 0 | 998 | 997 |
| 29 | 0 | 0 | 998 | 997 |
| 30 | 0 | 0 | 998 | 997 |

**折线图：**

**柱状图**

**部分实验结果截图：**

在第10天10：13所有患病学生均被送往医院



**2.第二次试验**

**表格：**

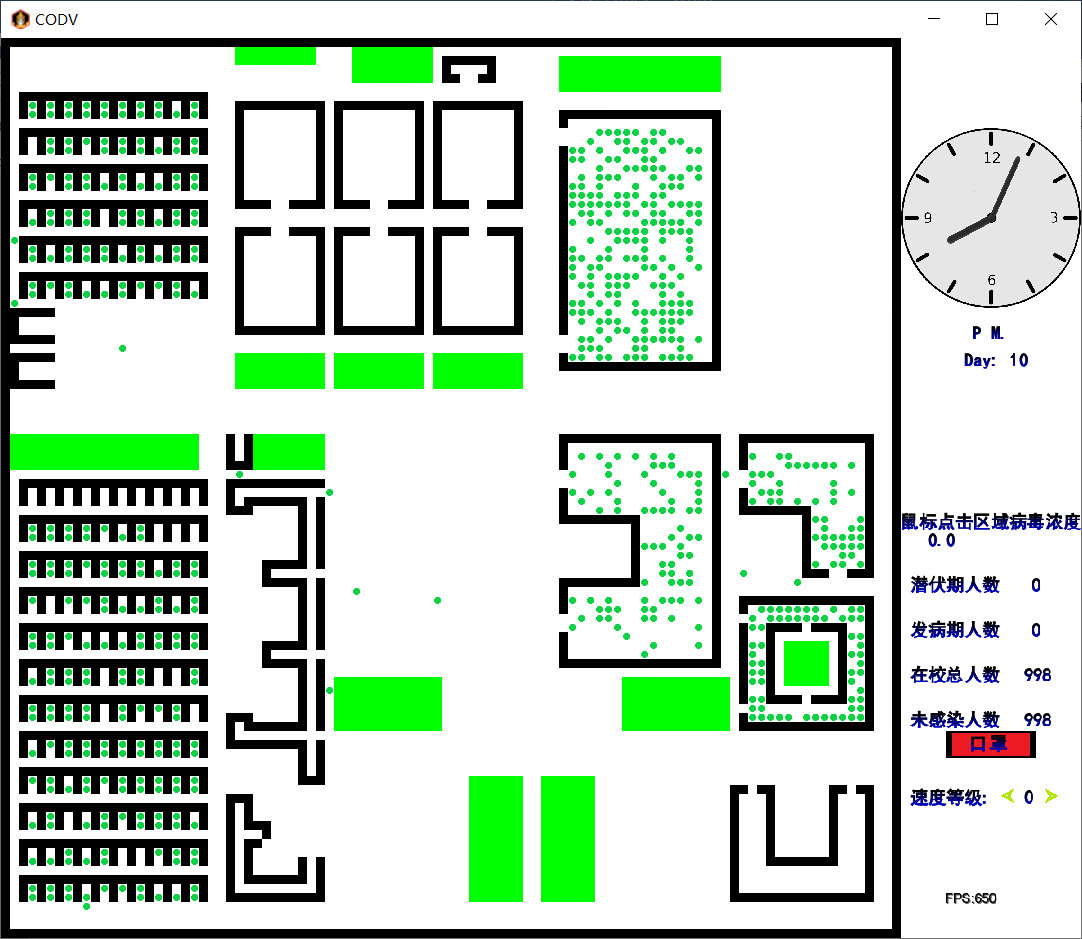
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 潜伏期人数 | 发病期人数 | 在校总人数 | 未感染人数 |
| 1 | 1 | 0 | 1000 | 999 |
| 2 | 1 | 0 | 1000 | 999 |
| 3 | 2 | 0 | 1000 | 998 |
| 4 | 2 | 0 | 1000 | 998 |
| 5 | 2 | 0 | 1000 | 998 |
| 6 | 1 | 1 | 999 | 998 |
| 7 | 1 | 0 | 999 | 998 |
| 8 | 1 | 0 | 999 | 998 |
| 9 | 1 | 0 | 999 | 998 |
| 10 | 0 | 1 | 999 | 998 |
| 11 | 0 | 0 | 998 | 998 |
| 12 | 0 | 0 | 998 | 998 |
| 13 | 0 | 0 | 998 | 998 |
| 14 | 0 | 0 | 998 | 998 |
| 15 | 0 | 0 | 998 | 998 |
| 16 | 0 | 0 | 998 | 998 |
| 17 | 0 | 0 | 998 | 998 |
| 18 | 0 | 0 | 998 | 998 |
| 19 | 0 | 0 | 998 | 998 |
| 20 | 0 | 0 | 998 | 998 |
| 21 | 0 | 0 | 998 | 998 |
| 22 | 0 | 0 | 998 | 998 |
| 23 | 0 | 0 | 998 | 998 |
| 24 | 0 | 0 | 998 | 998 |
| 25 | 0 | 0 | 998 | 998 |
| 26 | 0 | 0 | 998 | 998 |
| 27 | 0 | 0 | 998 | 998 |
| 28 | 0 | 0 | 998 | 998 |
| 29 | 0 | 0 | 998 | 998 |
| 30 | 0 | 0 | 998 | 998 |

**折线图：**

**柱状图**

**部分实验结果截图：**

第10天20：04所有患病学生均被送往医院



**三．源代码**

**1. codv.Actors: Clock.class**package codv.Actors;  
  
import codv.Codv;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.Batch;  
import com.badlogic.gdx.graphics.g2d.BitmapFont;  
import com.badlogic.gdx.graphics.g2d.Sprite;  
import com.badlogic.gdx.scenes.scene2d.Actor;  
  
public class Clock extends Actor {  
 Sprite clock;  
 Sprite hourHand;  
 Sprite minuteHand;  
 Codv codv;  
 BitmapFont font;  
  
 public Clock(Codv codv, int x, int y) {  
 this.codv = codv;  
 this.font = (BitmapFont)codv.manager.get("word.fnt");  
 this.setX((float)x);  
 this.setY((float)y);  
 this.setSize(200.0F, 200.0F);  
 this.clock = new Sprite((Texture)codv.manager.get("clock.png"));  
 this.clock.setSize(200.0F, 200.0F);  
 this.clock.setPosition(this.getX(), this.getY());  
 this.hourHand = new Sprite((Texture)codv.manager.get("hourHand.png"));  
 this.hourHand.setSize(8.0F, 60.0F);  
 this.hourHand.setPosition(this.getX() + this.getWidth() / 2.0F - 3.0F, this.getY() + this.getHeight() / 2.0F - 4.0F);  
 this.hourHand.setOrigin(4.0F, 4.0F);  
 this.minuteHand = new Sprite((Texture)codv.manager.get("hourHand.png"));  
 this.minuteHand.setSize(6.0F, 80.0F);  
 this.minuteHand.setPosition(this.getX() + this.getWidth() / 2.0F - 2.25F, this.getY() + this.getHeight() / 2.0F - 5.0F);  
 this.minuteHand.setOrigin(3.0F, 5.0F);  
 }  
  
 public void act(float delta) {  
 super.act(delta);  
 this.hourHand.setRotation(360.0F - (float)this.codv.gameScreen.gameStage.getTime() / 150.0F \* 30.0F);  
 this.minuteHand.setRotation(360.0F - (float)this.codv.gameScreen.gameStage.getTime() / 150.0F \* 360.0F);  
 }  
  
 public void draw(Batch batch, float parentAlpha) {  
 super.draw(batch, parentAlpha);  
 this.clock.draw(batch);  
 this.hourHand.draw(batch);  
 this.minuteHand.draw(batch);  
 if (this.codv.gameScreen.gameStage.getTime() < 1800) {  
 this.font.draw(batch, "A M.", this.getX() + 80.0F, this.getY() - 20.0F);  
 } else {  
 this.font.draw(batch, "P M.", this.getX() + 80.0F, this.getY() - 20.0F);  
 }  
  
 }  
}

**2. codv.Actors: Student.class**package codv.Actors;  
  
import codv.Codv;  
import codv.Point;  
import codv.Actors.Grids.Grid;  
import codv.Tasks.Task;  
import codv.Tasks.Eat.Breakfast;  
import codv.Tasks.Eat.Dinner;  
import codv.Tasks.Eat.Lunch;  
import codv.Tasks.Study.AfterNoonClass;  
import codv.Tasks.Study.GoToLibrary;  
import codv.Tasks.Study.MorningClass;  
import codv.Tasks.TouchFish.AdvanceSleep;  
import codv.Tasks.TouchFish.HangOut;  
import codv.Tasks.TouchFish.Sleep;  
import codv.Tasks.TouchFish.getMail;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.Batch;  
import com.badlogic.gdx.graphics.g2d.Sprite;  
import com.badlogic.gdx.scenes.scene2d.Actor;  
import java.util.Iterator;  
import java.util.LinkedList;  
import java.util.Random;  
  
public class Student extends Actor {  
 private static final float *incubationSpread* = 4.0E-4F;  
 private static final float *attackedSpread* = 0.002F;  
 private static final float *infectedProbablity* = 0.01F;  
 private static final Random *rand* = new Random();  
 private static final float *maskDefendRatio* = 0.6F;  
 private static final float *maskStopDiffusionRatio* = 0.2F;  
 public Codv codv;  
 public Grid[][] grids;  
 private Point position;  
 private Point dormPos;  
 private Point nowPos;  
 public static final int *diameter* = 8;  
 public Sprite red;  
 public Sprite orange;  
 public Sprite green;  
 private int status;  
 private int convertTime;  
 private LinkedList<Task> taskList;  
 private Task runningTask;  
 private Iterator<Task> taskIter;  
 private float last = 0.0F;  
 private int dfaId = 0;  
  
 public Student(Codv codv, Point position) {  
 this.codv = codv;  
 this.grids = codv.gameScreen.gameStage.grids;  
 this.position = position;  
 this.dormPos = position;  
 this.setPosition((float)position.x, (float)position.y);  
 this.red = new Sprite((Texture)codv.manager.get("redCircle.png"));  
 this.orange = new Sprite((Texture)codv.manager.get("orangeCircle.png"));  
 this.green = new Sprite((Texture)codv.manager.get("greenCircle.png"));  
 this.red.setSize(8.0F, 8.0F);  
 this.green.setSize(8.0F, 8.0F);  
 this.orange.setSize(8.0F, 8.0F);  
 this.taskList = new LinkedList();  
 this.status = 0;  
 }  
  
 private void workingDay() {  
 this.dfaId = 0;  
  
 do {  
 switch(this.dfaId) {  
 case -1:  
 this.taskList.add(new Sleep(this.codv, this.dormPos));  
 this.dfaId = 0;  
 break;  
 case 0:  
 this.taskList.add(new AdvanceSleep(this.codv, this.dormPos));  
 this.dfaId = 1;  
 break;  
 case 1:  
 this.taskList.add(new Breakfast(this.codv));  
 this.dfaId = 2;  
 break;  
 case 2:  
 this.taskList.add(new MorningClass(this.codv));  
 this.dfaId = 3;  
 break;  
 case 3:  
 this.taskList.add(new Lunch(this.codv));  
 this.dfaId = 4;  
 break;  
 case 4:  
 this.taskList.add(new AfterNoonClass(this.codv));  
 this.dfaId = 5;  
 break;  
 case 5:  
 this.taskList.add(new Dinner(this.codv));  
 if (*rand*.nextInt(10) < 3) {  
 this.dfaId = 6;  
 } else {  
 this.dfaId = 7;  
 }  
 break;  
 case 6:  
 this.taskList.add(new getMail(this.codv));  
 this.dfaId = 7;  
 break;  
 case 7:  
 int choice = *rand*.nextInt(10);  
 if (choice < 3) {  
 this.taskList.add(new AdvanceSleep(this.codv, this.dormPos));  
 this.dfaId = 0;  
 } else if (choice < 6) {  
 this.dfaId = 8;  
 } else if (choice < 9) {  
 this.dfaId = 9;  
 } else {  
 this.dfaId = 10;  
 }  
 break;  
 case 8:  
 this.taskList.add(new GoToLibrary(this.codv));  
 this.dfaId = -1;  
 break;  
 case 9:  
 this.taskList.add(new AfterNoonClass(this.codv));  
 this.dfaId = -1;  
 break;  
 case 10:  
 this.taskList.add(new HangOut(this.codv));  
 this.taskList.add(new HangOut(this.codv));  
 this.taskList.add(new HangOut(this.codv));  
 this.taskList.add(new HangOut(this.codv));  
 this.taskList.add(new AdvanceSleep(this.codv, this.dormPos));  
 this.dfaId = 0;  
 }  
 } while(this.dfaId != 0);  
  
 }  
  
 private void weekend() {  
 this.workingDay();  
 }  
  
 public void initTask() {  
 this.taskList.clear();  
 if (this.codv.gameScreen.gameStage.getDay() <= 5) {  
 this.workingDay();  
 } else {  
 this.weekend();  
 }  
  
 this.taskIter = this.taskList.iterator();  
 this.runningTask = (Task)this.taskIter.next();  
 }  
  
 private void update() {  
 int nowx = this.position.x / 10;  
 int nowy = this.position.y / 10;  
 }  
  
 public void getInfected() {  
 this.setZIndex(10999);  
 this.status = 1;  
 this.codv.addInfectedNumber();  
 this.codv.minusUninfectedNumber();  
 this.convertTime = *rand*.nextInt(25200) + 3600;  
 System.*out*.println("infected number = " + this.codv.getInfectedNumber());  
 }  
  
 public void getAttacked() {  
 this.setZIndex(11000);  
 this.status = 2;  
 this.convertTime = *rand*.nextInt(3600) + 3600;  
 this.codv.minusInfectedNumber();  
 this.codv.addAttackedNumber();  
 }  
  
 public void sendedToHospital() {  
 this.codv.minusAttackedNumber();  
 this.codv.minusStudentNumber();  
 this.remove();  
 }  
  
 private void spreadVirus(float increment) {  
 if (this.codv.getMasked()) {  
 this.grids[this.nowPos.x][this.nowPos.y].addVirusConcentration((double)(increment \* 0.2F));  
 } else {  
 this.grids[this.nowPos.x][this.nowPos.y].addVirusConcentration((double)increment);  
 }  
  
 }  
  
 private void calcInfected() {  
 if (this.codv.getMasked()) {  
 if ((double)*rand*.nextFloat() < 0.009999999776482582D \* this.grids[this.nowPos.x][this.nowPos.y].getVirusConcentration() \* 0.6000000238418579D) {  
 this.getInfected();  
 }  
 } else if ((double)*rand*.nextFloat() < 0.009999999776482582D \* this.grids[this.nowPos.x][this.nowPos.y].getVirusConcentration()) {  
 this.getInfected();  
 }  
  
 }  
  
 private void calcVirus() {  
 this.nowPos = new Point((int)this.getX() / 10, (int)this.getY() / 10);  
 if (this.status == 0) {  
 this.calcInfected();  
 } else if (this.status == 1) {  
 this.spreadVirus(4.0E-4F);  
 if (this.convertTime == 0) {  
 this.getAttacked();  
 }  
 } else if (this.status == 2) {  
 this.spreadVirus(0.002F);  
 if (this.convertTime == 0) {  
 this.sendedToHospital();  
 }  
 }  
  
 }  
  
 public void act(float delta) {  
 super.act(delta);  
 this.last += delta;  
 if (this.last > this.codv.gameScreen.gameStage.getThreshold()) {  
 this.last = 0.0F;  
 --this.convertTime;  
 if (this.runningTask == null) {  
 this.calcVirus();  
 return;  
 }  
  
 if (this.runningTask.isRunning()) {  
 this.position = this.runningTask.act();  
 } else if (this.runningTask.isFinish()) {  
 if (this.taskIter.hasNext()) {  
 this.runningTask = (Task)this.taskIter.next();  
 } else {  
 this.runningTask = null;  
 }  
 } else if (this.codv.gameScreen.gameStage.getTime() >= this.runningTask.getStartTime()) {  
 this.runningTask.startTask(this.position);  
 }  
  
 this.calcVirus();  
 } else {  
 this.setPosition(this.getX() + ((float)(this.position.x \* 10 + 1) - this.getX()) \* this.last / this.codv.gameScreen.gameStage.getThreshold(), this.getY() + ((float)(this.position.y \* 10 + 1) - this.getY()) \* this.last / this.codv.gameScreen.gameStage.getThreshold());  
 }  
  
 }  
  
 public void draw(Batch batch, float parentAlpha) {  
 super.draw(batch, parentAlpha);  
 if (this.status == 0) {  
 this.green.setPosition(this.getX(), this.getY());  
 this.green.draw(batch);  
 } else if (this.status == 1) {  
 this.orange.setPosition(this.getX(), this.getY());  
 this.orange.draw(batch);  
 } else if (this.status == 2) {  
 this.red.setPosition(this.getX(), this.getY());  
 this.red.draw(batch);  
 }  
  
 }  
}

**3.** **codv.Areas: Area.class**package codv.Areas;  
  
import codv.Point;  
import java.util.Random;  
  
public class Area {  
 protected Point leftBottom;  
 protected Point rightTop;  
 private int width;  
 private int height;  
 protected static Random *rand* = new Random();  
  
 public void init(Point leftBottom, Point rightTop) {  
 this.leftBottom = leftBottom;  
 this.rightTop = rightTop;  
 this.width = rightTop.x - leftBottom.x;  
 this.height = rightTop.y - leftBottom.y;  
 }  
  
 public Area() {  
 }  
  
 public Area(Point leftBottom, Point rightTop) {  
 this.init(leftBottom, rightTop);  
 }  
  
 public Point getRandomPoint() {  
 return new Point(this.leftBottom.x + *rand*.nextInt(this.width), this.leftBottom.y + *rand*.nextInt(this.height));  
 }  
}

**4.** **codv.Areas: Canteen.class**

package codv.Areas;  
  
import codv.Point;  
  
public class Canteen extends Area {  
 public Canteen() {  
 switch(rand.nextInt(4)) {  
 case 0:  
 this.init(new Point(27, 68), new Point(35, 78));  
 break;  
 case 1:  
 this.init(new Point(27, 82), new Point(35, 92));  
 break;  
 case 2:  
 this.init(new Point(38, 68), new Point(46, 78));  
 break;  
 case 3:  
 this.init(new Point(38, 82), new Point(46, 92));  
 }  
  
 }  
}

**5.** **codv.Areas: Classroom.class**

package codv.Areas;  
  
import codv.Point;  
  
public class ClassRoom extends Area {  
 public ClassRoom() {  
 switch(rand.nextInt(9)) {  
 case 0:  
 this.init(new Point(63, 47), new Point(78, 54));  
 break;  
 case 1:  
 this.init(new Point(63, 31), new Point(78, 38));  
 break;  
 case 2:  
 this.init(new Point(71, 39), new Point(78, 46));  
 break;  
 case 3:  
 this.init(new Point(83, 48), new Point(96, 54));  
 break;  
 case 4:  
 this.init(new Point(90, 41), new Point(96, 47));  
 break;  
 case 5:  
 this.init(new Point(83, 24), new Point(85, 37));  
 break;  
 case 6:  
 this.init(new Point(83, 35), new Point(96, 37));  
 break;  
 case 7:  
 this.init(new Point(83, 24), new Point(96, 25));  
 break;  
 case 8:  
 this.init(new Point(94, 24), new Point(96, 36));  
 }  
  
 }  
}

**6. codv.Areas: Dorm.class**

package codv.Areas;  
  
import codv.Point;  
  
public class Dorm extends Area {  
 public Dorm(Point point) {  
 this.leftBottom = point;  
 }  
  
 public Point getRandomPoint() {  
 return this.leftBottom;  
 }  
}

**7. codv.Areas: Library.class**

package codv.Areas;  
  
import codv.Point;  
  
public class Library extends Area {  
 public Library() {  
 this.init(new Point(63, 64), new Point(78, 90));  
 }  
}

**8. codv.Areas: PlayGround.class**

package codv.Areas;  
  
import codv.Point;  
  
public class PlayGround extends Area {  
 public PlayGround() {  
 this.init(new Point(36, 30), new Point(57, 53));  
 }  
}

**9. codv.Areas: postOffice.class**

package codv.Areas;  
  
import codv.Point;  
  
public class postOffice extends Area {  
 public postOffice() {  
 if (rand.nextInt(2) < 1) {  
 this.init(new Point(2, 67), new Point(5, 69));  
 } else {  
 this.init(new Point(2, 62), new Point(5, 64));  
 }  
  
 }  
}

**10.** **codv.Mybuttons: MaskButton.class**

package codv.Mybuttons;  
  
import codv.Codv;  
import com.badlogic.gdx.audio.Sound;  
import com.badlogic.gdx.graphics.g2d.Batch;  
import com.badlogic.gdx.scenes.scene2d.InputEvent;  
import com.badlogic.gdx.scenes.scene2d.utils.ClickListener;  
  
public class MaskButton extends MyButton {  
 Sound select;  
  
 public MaskButton(final Codv codv, int x, int y, int width, int height) {  
 super(codv, x, y, width, height);  
 this.select = (Sound)codv.manager.get("select.wav");  
 this.addListener(new ClickListener() {  
 public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {  
 MaskButton.this.clickedStatus = !MaskButton.this.clickedStatus;  
 codv.reverseMasked();  
 MaskButton.this.select.play();  
 return super.touchDown(event, x, y, pointer, button);  
 }  
  
 public void touchUp(InputEvent event, float x, float y, int pointer, int button) {  
 super.touchUp(event, x, y, pointer, button);  
 }  
 });  
 }  
  
 public void draw(Batch batch, float parentAlpha) {  
 if (!this.clickedStatus) {  
 this.white.setX(this.getX());  
 this.white.setY(this.getY());  
 this.white.draw(batch);  
 } else {  
 this.red.setX(this.getX());  
 this.red.setY(this.getY());  
 this.red.draw(batch);  
 }  
  
 this.font.draw(batch, "口罩", this.getX() + 28.0F, this.getY() + 24.0F);  
 }  
}

**11.** **codv.Mybuttons: MyButton.class**

package codv.Mybuttons;  
  
import codv.Codv;  
import com.badlogic.gdx.graphics.Color;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.Batch;  
import com.badlogic.gdx.graphics.g2d.BitmapFont;  
import com.badlogic.gdx.graphics.g2d.Sprite;  
import com.badlogic.gdx.scenes.scene2d.Actor;  
  
public class MyButton extends Actor {  
 BitmapFont font;  
 Codv codv;  
 Sprite red;  
 Sprite black;  
 Sprite white;  
 boolean clickedStatus = false;  
  
 MyButton(Codv codv, int x, int y, int width, int height) {  
 this.setX((float)x);  
 this.setY((float)y);  
 this.setWidth((float)width);  
 this.setHeight((float)height);  
 this.codv = codv;  
 this.font = (BitmapFont)codv.manager.get("word.fnt");  
 this.red = new Sprite((Texture)codv.manager.get("red.png"));  
 this.white = new Sprite((Texture)codv.manager.get("white.png"));  
 this.black = new Sprite((Texture)codv.manager.get("black.png"));  
 this.red.setSize((float)width, (float)height);  
 this.white.setSize((float)width, (float)height);  
 this.black.setSize((float)width, (float)height);  
 }  
  
 public void draw(Batch batch, float parentAlpha) {  
 if (!this.clickedStatus) {  
 this.white.setX(this.getX());  
 this.white.setY(this.getY());  
 this.white.draw(batch);  
 } else {  
 this.red.setX(this.getX());  
 this.red.setY(this.getY());  
 this.red.draw(batch);  
 }  
  
 this.font.setColor(Color.BLACK);  
 this.font.draw(batch, "Start", this.getX() + 16.0F, this.getY() + this.getHeight());  
 super.draw(batch, parentAlpha);  
 }  
}

**12.** **codv.Navagate: NavagatePoint.class**

package codv.Navagate;  
import codv.Point;  
  
public class NavigatePoint extends Point {  
 int dis;  
 NavigatePoint parent;  
  
 NavigatePoint(int x, int y, int x1, int y1, NavigatePoint parent) {  
 super(x, y);  
 this.dis = Math.*abs*(x1 - x) + Math.*abs*(y1 - y);  
 this.parent = parent;  
 }  
  
 NavigatePoint(Point a, Point target, NavigatePoint parent) {  
 super(a);  
 this.dis = Math.*abs*(target.x - a.x) + Math.*abs*(target.y - a.y);  
 this.parent = parent;  
 }  
  
 NavigatePoint(Point a, Point target) {  
 super(a);  
 this.dis = Math.*abs*(target.x - a.x) + Math.*abs*(target.y - a.y);  
 }  
}

**13. codv.Navagate: Path.class**

package codv.Navagate;  
  
import codv.Point;  
import codv.Actors.Grids.Grid;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.Iterator;  
import java.util.LinkedList;  
import java.util.List;  
import java.util.PriorityQueue;  
  
public class Path {  
 private Grid[][] grids;  
 private LinkedList<Point> path;  
 private Iterator<Point> iter;  
 private static final Point[] *direction* = new Point[]{new Point(0, 1), new Point(1, 0), new Point(0, -1), new Point(-1, 0)};  
 private Point st;  
 private Point ed;  
  
 public Path(Grid[][] grids, Point st, Point ed) {  
 this.grids = grids;  
 this.path = new LinkedList();  
 this.st = st;  
 this.ed = ed;  
 this.aStar(new Point(st), new Point(ed));  
 }  
  
 private void aStar(Point start, Point end) {  
 ++Grid.*nowId*;  
 Comparator<NavigatePoint> cPoint = new Comparator<NavigatePoint>() {  
 public int compare(NavigatePoint o1, NavigatePoint o2) {  
 return o1.dis - o2.dis;  
 }  
 };  
 PriorityQueue<NavigatePoint> heap = new PriorityQueue(cPoint);  
 heap.add(new NavigatePoint(start, end, new NavigatePoint(start, end)));  
 int var6 = 0;  
  
 while(true) {  
 ++var6;  
 NavigatePoint now = (NavigatePoint)heap.poll();  
 if (now.equal(end)) {  
 while(true) {  
 this.path.addFirst(new Point(now.x, now.y));  
 if (now.equal(now.parent)) {  
 this.iter = this.path.iterator();  
 heap.clear();  
 return;  
 }  
  
 now = now.parent;  
 }  
 }  
  
 List<Integer> random = new ArrayList();  
 random.add(0);  
 random.add(1);  
 random.add(2);  
 random.add(3);  
 Collections.*shuffle*(random);  
  
 for(int i = 0; i < 4; ++i) {  
 Point nextPoint = now.add(*direction*[(Integer)random.get(i)]);  
 if (nextPoint.islegal() && this.grids[nextPoint.x][nextPoint.y].passable() && this.grids[nextPoint.x][nextPoint.y].id != Grid.*nowId*) {  
 heap.add(new NavigatePoint(nextPoint, end, now));  
 this.grids[nextPoint.x][nextPoint.y].id = Grid.*nowId*;  
 }  
 }  
 }  
 }  
  
 public Point getNextPoint() {  
 return this.iter.hasNext() ? (Point)this.iter.next() : new Point(-1, -1);  
 }  
}

**14. codv.Tasks.Eat: Breakfast.class**

package codv.Tasks.Eat;  
  
import codv.Codv;  
  
public class Breakfast extends Eat {  
 public Breakfast(Codv codv) {  
 super(codv);  
 this.init(900 + *rand*.nextInt(150));  
 }  
}

**15. codv.Tasks.Eat: Dinner.class**

package codv.Tasks.Eat;  
  
import codv.Codv;  
  
public class Dinner extends Eat {  
 public Dinner(Codv codv) {  
 super(codv);  
 this.init(2400 + *rand*.nextInt(300));  
 }  
}

**16. codv.Tasks.Eat: Eat.class**

package codv.Tasks.Eat;  
  
import codv.Codv;  
import codv.Areas.Canteen;  
import codv.Tasks.Task;  
import java.util.Random;  
  
public class Eat extends Task {  
 protected static Random *rand* = new Random();  
  
 public void init(int startTime) {  
 this.init(new Canteen(), startTime);  
 }  
  
 public Eat(Codv codv) {  
 super(codv);  
 }  
}

**17. codv.Tasks.Eat: Lunch.class**

package codv.Tasks.Eat;  
  
import codv.Codv;  
  
public class Lunch extends Eat {  
 public Lunch(Codv codv) {  
 super(codv);  
 this.init(1650 + *rand*.nextInt(300));  
 }  
}

**18.** **codv.Tasks.Study: AfterNoonClass.class**

package codv.Tasks.Study;  
  
import codv.Codv;  
  
public class AfterNoonClass extends GoToClass {  
 public AfterNoonClass(Codv codv) {  
 super(codv);  
 this.init(1950 + *rand*.nextInt(300));  
 }  
}

**19. codv.Tasks.Study: GoToClass.class**

package codv.Tasks.Study;  
  
import codv.Codv;  
import codv.Areas.ClassRoom;  
import codv.Tasks.Task;  
import java.util.Random;  
  
public class GoToClass extends Task {  
 protected static Random *rand* = new Random();  
  
 public void init(int startTime) {  
 this.init(new ClassRoom(), startTime);  
 }  
  
 public GoToClass(Codv codv) {  
 super(codv);  
 }  
}

**20. codv.Tasks.Study: GoToLibrary.class**

package codv.Tasks.Study;  
  
import codv.Codv;  
import codv.Areas.Library;  
import codv.Tasks.Task;  
  
public class GoToLibrary extends Task {  
 public GoToLibrary(Codv codv) {  
 super(codv);  
 this.init(new Library(), 0);  
 }  
}

**21. codv.Tasks.Study: MorningClass.class**

package codv.Tasks.Study;  
  
import codv.Codv;  
  
public class MorningClass extends GoToClass {  
 public MorningClass(Codv codv) {  
 super(codv);  
 this.init(1050 + *rand*.nextInt(450));  
 }  
}

**22.** **codv.Tasks.TouchFish: AdvanceSleep.class**

package codv.Tasks.TouchFish;  
  
import codv.Codv;  
import codv.Point;  
import codv.Areas.Dorm;  
import codv.Tasks.Task;  
import java.util.Random;  
  
public class AdvanceSleep extends Task {  
 private static Random *rand* = new Random();  
  
 public AdvanceSleep(Codv codv, Point dormPos) {  
 super(codv);  
 this.init(new Dorm(dormPos), 0);  
 }  
}

**23.** **codv.Tasks.TouchFish: getMail.class**

package codv.Tasks.TouchFish;  
  
import codv.Codv;  
import codv.Areas.postOffice;  
import codv.Tasks.Task;  
  
public class getMail extends Task {  
 public getMail(Codv codv) {  
 super(codv);  
 this.init(new postOffice(), 0);  
 }  
}

**24.** **codv.Tasks.TouchFish: HangOut.class**

package codv.Tasks.TouchFish;  
  
import codv.Codv;  
import codv.Areas.PlayGround;  
import codv.Tasks.Task;  
  
public class HangOut extends Task {  
 public HangOut(Codv codv) {  
 super(codv);  
 this.init(new PlayGround(), 0);  
 }  
}

**25.** **codv.Tasks.TouchFish: Sleep.class**

package codv.Tasks.TouchFish;  
  
import codv.Codv;  
import codv.Point;  
import codv.Areas.Dorm;  
import codv.Tasks.Task;  
import java.util.Random;  
  
public class Sleep extends Task {  
 private static Random *rand* = new Random();  
  
 public Sleep(Codv codv, Point dormPos) {  
 super(codv);  
 this.init(new Dorm(dormPos), 3000 + *rand*.nextInt(450));  
 }  
}

**26.** **codv.Tasks: DoNothing.class**

package codv.Tasks;  
  
import codv.Point;  
import java.util.Random;  
  
public class DoNothing {  
 private static Random *rand* = new Random();  
  
 public DoNothing() {  
 }  
  
 public static Point act(Point nowPos) {  
 switch(*rand*.nextInt(4)) {  
 case 0:  
 return nowPos.add(new Point(1, 0));  
 case 1:  
 return nowPos.add(new Point(-1, 0));  
 case 2:  
 return nowPos.add(new Point(0, 1));  
 default:  
 return nowPos.add(new Point(0, -1));  
 }  
 }  
}

**27.** **codv.Tasks: Task.class**

package codv.Tasks;  
  
import codv.Codv;  
import codv.Point;  
import codv.Areas.Area;  
import codv.Navagate.Path;  
  
public class Task {  
 private Codv codv;  
 private Point endPoint;  
 private Point startPoint;  
 private Area startArea;  
 private Area endArea;  
 private int startTime;  
 private Path path;  
 private boolean running = false;  
 private boolean finish = false;  
  
 public void init(Area endArea, int startTime) {  
 this.endArea = endArea;  
 this.startTime = startTime;  
 }  
  
 public Task(Codv codv) {  
 this.codv = codv;  
 }  
  
 public Task(Codv codv, Area endArea, int startTime) {  
 this.codv = codv;  
 this.startArea = this.startArea;  
 this.init(endArea, startTime);  
 }  
  
 public boolean isRunning() {  
 return this.running;  
 }  
  
 public boolean isFinish() {  
 return this.finish;  
 }  
  
 public int getStartTime() {  
 return this.startTime;  
 }  
  
 public void startTask(Point nowPos) {  
 this.running = true;  
 this.startPoint = nowPos;  
 this.endPoint = this.endArea.getRandomPoint();  
 this.path = new Path(this.codv.gameScreen.gameStage.grids, this.startPoint, this.endPoint);  
 }  
  
 public Point act() {  
 if (!this.running) {  
 return new Point(-1, -1);  
 } else {  
 Point nextPoint = this.path.getNextPoint();  
 if (nextPoint.equal(this.endPoint)) {  
 this.running = false;  
 this.finish = true;  
 }  
  
 return nextPoint;  
 }  
 }  
}

**28. codv: Codv.class**

package codv;  
  
import codv.Tools.PointerSimulator;  
import com.badlogic.gdx.Game;  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.assets.AssetManager;  
import com.badlogic.gdx.audio.Sound;  
import com.badlogic.gdx.graphics.Texture;  
import com.badlogic.gdx.graphics.g2d.BitmapFont;  
  
public class Codv extends Game {  
 public GameScreen gameScreen;  
 public AssetManager manager;  
 public PointerSimulator<Integer> infectedNumber;  
 public PointerSimulator<Integer> uninfectedNumber;  
 public PointerSimulator<Integer> attackedNumber;  
 public PointerSimulator<Integer> studentNumber;  
 public PointerSimulator<Integer> dayNumber;  
 public PointerSimulator<Float> virusConcentrationNumber;  
 public PointerSimulator<Integer> speedNumber;  
 private boolean masked = false;  
  
 public Codv() {  
 }  
  
 public int getInfectedNumber() {  
 return (Integer)this.infectedNumber.value;  
 }  
  
 public void setInfectedNumber(int infectedNumber) {  
 this.infectedNumber.value = infectedNumber;  
 }  
  
 public void addInfectedNumber() {  
 PointerSimulator var1 = this.infectedNumber;  
 Integer var2 = (Integer)var1.value;  
 var1.value = (Integer)var1.value + 1;  
 }  
  
 public void minusInfectedNumber() {  
 PointerSimulator var1 = this.infectedNumber;  
 Integer var2 = (Integer)var1.value;  
 var1.value = (Integer)var1.value - 1;  
 }  
  
 public int getAttackedNumber() {  
 return (Integer)this.attackedNumber.value;  
 }  
  
 public void setAttackedNumber(int attackedNumber) {  
 this.attackedNumber.value = attackedNumber;  
 }  
  
 public void addAttackedNumber() {  
 PointerSimulator var1 = this.attackedNumber;  
 Integer var2 = (Integer)var1.value;  
 var1.value = (Integer)var1.value + 1;  
 }  
  
 public void minusAttackedNumber() {  
 PointerSimulator var1 = this.attackedNumber;  
 Integer var2 = (Integer)var1.value;  
 var1.value = (Integer)var1.value - 1;  
 }  
  
 public boolean getMasked() {  
 return this.masked;  
 }  
  
 public void setMasked(boolean newStatus) {  
 this.masked = newStatus;  
 }  
  
 public void reverseMasked() {  
 this.masked = !this.masked;  
 }  
  
 public void loadResources() {  
 this.manager = new AssetManager();  
 this.manager.load("block.png", Texture.class);  
 this.manager.load("greenCircle.png", Texture.class);  
 this.manager.load("redCircle.png", Texture.class);  
 this.manager.load("orangeCircle.png", Texture.class);  
 this.manager.load("word.fnt", BitmapFont.class);  
 this.manager.load("black.png", Texture.class);  
 this.manager.load("white.png", Texture.class);  
 this.manager.load("red.png", Texture.class);  
 this.manager.load("clock.png", Texture.class);  
 this.manager.load("hourHand.png", Texture.class);  
 this.manager.load("arrow.png", Texture.class);  
 this.manager.load("arrowTouched.png", Texture.class);  
 this.manager.load("select.wav", Sound.class);  
 this.manager.load("notOk.wav", Sound.class);  
 this.manager.finishLoading();  
 }  
  
 public PointerSimulator<Integer> getStudentNumber() {  
 return this.studentNumber;  
 }  
  
 public void setStudentNumber(PointerSimulator<Integer> studentNumber) {  
 this.studentNumber = studentNumber;  
 }  
  
 public void minusStudentNumber() {  
 PointerSimulator var1 = this.studentNumber;  
 Integer var2 = (Integer)var1.value;  
 var1.value = (Integer)var1.value - 1;  
 }  
  
 public void minusUninfectedNumber() {  
 PointerSimulator var1 = this.uninfectedNumber;  
 Integer var2 = (Integer)var1.value;  
 var1.value = (Integer)var1.value - 1;  
 }  
  
 public void create() {  
 this.infectedNumber = new PointerSimulator(0);  
 this.attackedNumber = new PointerSimulator(0);  
 this.studentNumber = new PointerSimulator(1000);  
 this.uninfectedNumber = new PointerSimulator(1000);  
 this.dayNumber = new PointerSimulator(1);  
 this.virusConcentrationNumber = new PointerSimulator(0.0F);  
 this.speedNumber = new PointerSimulator(4);  
 this.loadResources();  
 this.gameScreen = new GameScreen(this);  
 this.gameScreen.init();  
 this.setScreen(this.gameScreen);  
 Gdx.*input*.setInputProcessor(this.gameScreen.gameStage);  
 }  
}

**29. codv: GameScreen.class**

package codv;  
  
import com.badlogic.gdx.Gdx;  
import com.badlogic.gdx.ScreenAdapter;  
import com.badlogic.gdx.graphics.Color;  
import com.badlogic.gdx.graphics.g2d.BitmapFont;  
import com.badlogic.gdx.graphics.g2d.SpriteBatch;  
import com.badlogic.gdx.utils.TimeUtils;  
  
public class GameScreen extends ScreenAdapter {  
 public GameStage gameStage;  
 public Codv codv;  
 public SpriteBatch batch;  
 public int FPS;  
 private int nowFPS;  
 private long lastFPS;  
 private BitmapFont font;  
  
 public GameScreen(Codv codv) {  
 this.codv = codv;  
 }  
  
 public void init() {  
 this.gameStage = new GameStage(this.codv);  
 this.gameStage.init();  
 this.font = new BitmapFont();  
 this.batch = new SpriteBatch();  
 Gdx.*input*.setInputProcessor(this.gameStage);  
 this.font.setColor(Color.*BLACK*);  
 }  
  
 public void render(float delta) {  
 super.render(delta);  
 Gdx.*gl*.glClearColor(1.0F, 1.0F, 1.0F, 1.0F);  
 Gdx.*gl*.glClear(16384);  
 if (TimeUtils.*nanoTime*() - this.lastFPS >= 1000000000L) {  
 this.lastFPS = TimeUtils.*nanoTime*();  
 this.FPS = this.nowFPS;  
 this.nowFPS = 0;  
 }  
  
 ++this.nowFPS;  
 this.batch.begin();  
 if (!this.gameStage.pause) {  
 this.gameStage.act();  
 }  
  
 this.gameStage.draw();  
 this.font.draw(this.batch, "FPS:" + this.FPS, 1050.0F, 50.0F);  
 this.batch.end();  
 }  
  
 public void resize(int width, int height) {  
 super.resize(width, height);  
 this.gameStage.getViewport().update(width, height);  
 this.gameStage.getCamera().update();  
 }  
}

**30. codv: GameStage.class**

package codv;  
  
import codv.Actors.Clock;  
import codv.Actors.Student;  
import codv.Actors.Grids.GreenBelt;  
import codv.Actors.Grids.Grid;  
import codv.Actors.Grids.Indoor;  
import codv.Actors.Grids.Space;  
import codv.Actors.Grids.Wall;  
import codv.Mybuttons.MaskButton;  
import codv.Tools.FloatMonitor;  
import codv.Tools.Modifier;  
import codv.Tools.Monitor;  
import codv.Tools.PointerSimulator;  
import codv.Tools.Sentences;  
import codv.Tools.getDorm;  
import com.badlogic.gdx.scenes.scene2d.Stage;  
import com.badlogic.gdx.utils.viewport.StretchViewport;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Random;  
  
public class GameStage extends Stage {  
 private static Random *rand* = new Random();  
 public Codv codv;  
 public Grid[][] grids;  
 public Student[] students;  
 private int time = 0;  
 private int day = 1;  
 private int week = 1;  
 private float last = 0.0F;  
 private float threshold = 0.016F;  
 public boolean pause = false;  
 private boolean action = false;  
 private Monitor infectdNumberMonitor;  
 private Monitor attackedNumberMonitor;  
 private Monitor uninfectedNumberMonitor;  
 private Monitor virusConcentrationMonitor;  
 private Monitor dayMonitor;  
 private Monitor studentNumberMonitor;  
  
 public GameStage(Codv codv) {  
 super(new StretchViewport(1200.0F, 1000.0F));  
 this.codv = codv;  
 }  
  
 public int getTime() {  
 return this.time;  
 }  
  
 public int getDay() {  
 return this.day;  
 }  
  
 private void setControllerGroup(String string, int x, int y, int width, int height, PointerSimulator<Integer> target) {  
 this.addActor(new Sentences(this.codv, string, x - string.length() \* 20, y + 14));  
 this.addActor(new Modifier(this.codv, (int)((double)x + 50.0D \* (double)width / 16.0D), y, width, height, 1, target));  
 this.addActor(new Modifier(this.codv, x, y, width, height, -1, target));  
 this.addActor(new Monitor(this.codv, target, x + 22, y + 14));  
 }  
  
 private void menuInit() {  
 int x = 1140;  
 this.addActor(new Sentences(this.codv, "潜伏期人数", x - 130, 400));  
 this.infectdNumberMonitor = new Monitor(this.codv, this.codv.infectedNumber, x, 400);  
 this.addActor(new Sentences(this.codv, "发病期人数", x - 130, 350));  
 this.attackedNumberMonitor = new Monitor(this.codv, this.codv.attackedNumber, x, 350);  
 this.addActor(new Sentences(this.codv, "鼠标点击区域病毒浓度:", 1000, 470));  
 this.virusConcentrationMonitor = new FloatMonitor(this.codv, this.codv.virusConcentrationNumber, 1030, 450);  
 this.addActor(new Sentences(this.codv, "Day:", 1070, 650));  
 this.dayMonitor = new Monitor(this.codv, this.codv.dayNumber, 1120, 650);  
 this.addActor(new Sentences(this.codv, "在校总人数", x - 130, 300));  
 this.studentNumberMonitor = new Monitor(this.codv, this.codv.studentNumber, x, 300);  
 this.addActor(new Sentences(this.codv, "未感染人数", x - 130, 250));  
 this.uninfectedNumberMonitor = new Monitor(this.codv, this.codv.uninfectedNumber, x, 250);  
 this.setControllerGroup("速度等级:", x - 30, 150, 16, 16, this.codv.speedNumber);  
 this.addActor(this.uninfectedNumberMonitor);  
 this.addActor(this.studentNumberMonitor);  
 this.addActor(this.dayMonitor);  
 this.addActor(this.virusConcentrationMonitor);  
 this.addActor(this.infectdNumberMonitor);  
 this.addActor(this.attackedNumberMonitor);  
 this.addActor(new Clock(this.codv, 1000, 700));  
 this.addActor(new MaskButton(this.codv, 1050, 200, 100, 30));  
 }  
  
 private void getMapFromDataBase() {  
 String DB\_URL = "jdbc:mysql://116.62.52.154:3306/map?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";  
 String USER = "root";  
 String PASS = "123456";  
 Connection conn = null;  
 Statement stmt = null;  
  
 try {  
 System.*out*.println("连接数据库...");  
 conn = DriverManager.*getConnection*("jdbc:mysql://116.62.52.154:3306/map?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC", "root", "123456");  
 System.*out*.println(" 实例化Statement对象...");  
 String sql = "select id, type FROM MAP ";  
 stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(sql);  
 int tot = false;  
  
 for(int j = 99; j >= 0; --j) {  
 for(int i = 0; i < 100; ++i) {  
 if (rs.next()) {  
 int id = rs.getInt("id");  
 int type = rs.getInt("type");  
 switch(type) {  
 case 0:  
 this.grids[i][j] = new Space(this.codv, this.grids, new Point(i \* 10, j \* 10));  
 break;  
 case 1:  
 this.grids[i][j] = new Wall(this.codv, this.grids, new Point(i \* 10, j \* 10));  
 break;  
 case 2:  
 this.grids[i][j] = new Indoor(this.codv, this.grids, new Point(i \* 10, j \* 10));  
 break;  
 case 3:  
 this.grids[i][j] = new GreenBelt(this.codv, this.grids, new Point(i \* 10, j \* 10));  
 }  
  
 this.addActor(this.grids[i][j]);  
 this.grids[i][j].setZIndex(0);  
 }  
 }  
 }  
  
 rs.close();  
 stmt.close();  
 conn.close();  
 } catch (SQLException var28) {  
 var28.printStackTrace();  
 } catch (Exception var29) {  
 var29.printStackTrace();  
 } finally {  
 try {  
 if (stmt != null) {  
 stmt.close();  
 }  
 } catch (SQLException var27) {  
 }  
  
 try {  
 if (conn != null) {  
 conn.close();  
 }  
 } catch (SQLException var26) {  
 var26.printStackTrace();  
 }  
  
 }  
  
 }  
  
 public void init() {  
 this.grids = new Grid[100][100];  
 this.getMapFromDataBase();  
 this.students = new Student[1000];  
  
 for(int i = 0; i < 1000; ++i) {  
 this.students[i] = new Student(this.codv, new getDorm(i));  
 this.addActor(this.students[i]);  
 this.students[i].setZIndex(10000);  
 }  
  
 this.students[*rand*.nextInt(1)].getInfected();  
 this.menuInit();  
 }  
  
 public float getThreshold() {  
 return this.threshold;  
 }  
  
 public void mulThreshold(float mi) {  
 this.threshold \*= mi;  
 }  
  
 public boolean getAction() {  
 return this.action;  
 }  
  
 public void act(float delta) {  
 super.act(delta);  
 this.last += delta;  
 if (this.time == 0) {  
 for(int i = 0; i < 1000; ++i) {  
 this.students[i].initTask();  
 }  
 }  
  
 if (this.last > this.threshold) {  
 this.action = true;  
 this.last = 0.0F;  
 ++this.time;  
 if (this.time % 150 == 0) {  
 System.*out*.println(this.time / 150);  
 }  
  
 if (this.time >= 3600) {  
 this.time = 0;  
 PointerSimulator var3 = this.codv.dayNumber;  
 var3.value = (Integer)var3.value + 1;  
 }  
 } else {  
 this.action = false;  
 }  
  
 }  
}

**31. codv: Point.class**

package codv;  
  
public class Point {  
 public int x;  
 public int y;  
  
 public Point(int x, int y) {  
 this.x = x;  
 this.y = y;  
 }  
  
 public Point() {  
 }  
  
 public Point(Point rhs) {  
 this.x = rhs.x;  
 this.y = rhs.y;  
 }  
  
 public boolean equal(Point rhs) {  
 return this.x == rhs.x && this.y == rhs.y;  
 }  
  
 public Point add(Point rhs) {  
 return new Point(this.x + rhs.x, this.y + rhs.y);  
 }  
  
 public Point minus(Point rhs) {  
 return new Point(this.x - rhs.x, this.y - rhs.y);  
 }  
  
 public boolean islegal() {  
 return this.x >= 0 && this.x < 100 && this.y >= 0 && this.y < 100;  
 }  
  
 public int getdis(Point rhs) {  
 return Math.*abs*(this.x - rhs.x) + Math.*abs*(this.y - rhs.y);  
 }  
  
 public void print() {  
 System.*out*.println("x = " + this.x + " y =" + this.y);  
 }  
}

**32.** **codv.lwjgl3: Lwjgl3Launcher.class**

package codv.lwjgl3;  
  
import codv.Codv;  
import com.badlogic.gdx.backends.lwjgl3.Lwjgl3Application;  
import com.badlogic.gdx.backends.lwjgl3.Lwjgl3ApplicationConfiguration;  
  
public class Lwjgl3Launcher {  
 static final String *JDBC\_DRIVER* = "com.mysql.cj.jdbc.Driver";  
  
 public Lwjgl3Launcher() {  
 }  
  
 public static void main(String[] args) {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 } catch (ClassNotFoundException var2) {  
 var2.printStackTrace();  
 }  
  
 *createApplication*();  
 }  
  
 private static Lwjgl3Application createApplication() {  
 return new Lwjgl3Application(new Codv(), *getDefaultConfiguration*());  
 }  
  
 private static Lwjgl3ApplicationConfiguration getDefaultConfiguration() {  
 Lwjgl3ApplicationConfiguration configuration = new Lwjgl3ApplicationConfiguration();  
 configuration.setTitle("CODV");  
 configuration.setWindowedMode(1080, 900);  
 configuration.useVsync(false);  
 configuration.setWindowIcon(new String[]{"libgdx128.png", "libgdx64.png", "libgdx32.png", "libgdx16.png"});  
 return configuration;  
 }  
}