Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Лабораторная работа №4 по дисциплине  
«Технология программирования» на тему

***Многопотоковые приложения***

Факультет: АВТФ Преподаватель:

Группа: АВТ-810 Михайленко Д. А.

Студенты: Степаненко Владислав

Новосибирск

2020

#### Практические задания

1. Изучить особенности реализации и работы потоков в Java, управлением приоритетами потоков и синхронизацией потоков.
2. Доработать программу, созданную в лабораторной работу № 3:
3. создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;
4. реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;
5. синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;
6. добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов. Реализовать через засыпание/пробуждение потоков (методы wait() и notify());
7. добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

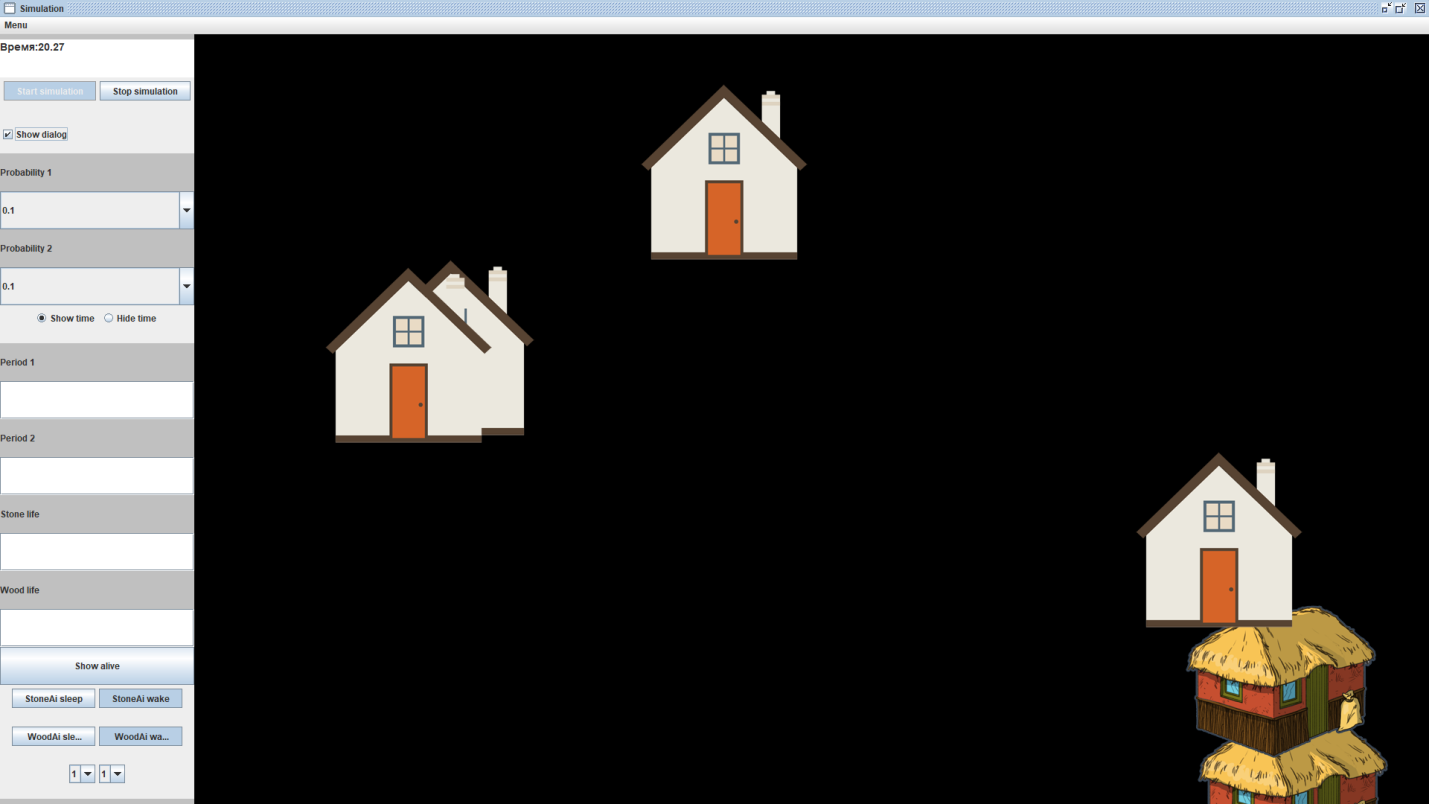
***Вариант 7***

1. Капитальные дома двигаются (в городах будущего и не такое возможно) в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.
2. Деревянные дома после генерации начинают двигаться в нижнюю правую четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке w/2;h/2, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.

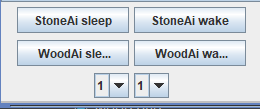
**Ход работы:**

Создан класс BaseAI наследник Thread, от него созданы потомки StoneAi и WoodAi. Они обновляют значения координат объектов в зависимости от поведения последних.

(конечная точка – случайные координаты в определенной области, находится вектор соединяющий начальную и конечную точку движения, считается “скорость” которая раскладывается на составляющие x и y)



Добавлены комбобоксы для выставления приоритета потокам, кнопки для вкл\откл AI.



**Листинг программы(основные изменения):**

public class StoneAI extends BaseAI  
{  
 Habitat h;  
 StoneAI(Habitat habitat)  
 {  
 h=habitat;  
 }  
 public void run() {  
 double speed = 1000;  
 while (true) {  
 if (working) {  
 for (int i = 0; i < h.houseSingleton.houseList.size() - 1; i++) {  
 if (h.houseSingleton.houseList.get(i) instanceof Stone) {  
 double x1 = h.houseSingleton.houseList.get(i).x;  
 double y1 = h.houseSingleton.houseList.get(i).y;  
 double x2 = (h.ui.simulation\_panel.getWidth() / 2) \* Math.*random*();  
 double y2 = (h.ui.simulation\_panel.getHeight() / 2) \* Math.*random*();  
 double a = x2 - x1;  
 double b = y2 - y1;  
 double c = a \* a + b \* b;  
 double vx = (speed \* a) / c;  
 h.houseSingleton.houseList.get(i).setVx(vx);//System.out.println(vx);  
 double vy = (speed \* b) / c;  
 h.houseSingleton.houseList.get(i).setVy(vy);//System.out.println(vy);  
 int tact = h.houseSingleton.houseList.get(i).getTact();  
 int lasttact = (int) (c / speed);  
 h.houseSingleton.houseList.get(i).setLasttact(lasttact);  
 if (h.houseSingleton.houseList.get(i).getTact() < h.houseSingleton.houseList.get(i).getLasttact()) {  
 tact++;  
 h.houseSingleton.houseList.get(i).setTact(tact);  
 x1 = x1 + h.houseSingleton.houseList.get(i).getVx();  
 y1 = y1 + h.houseSingleton.houseList.get(i).getVy();  
  
 h.houseSingleton.houseList.get(i).setX(x1);  
 h.houseSingleton.houseList.get(i).setY(y1);  
 }  
 }  
 }  
 try {  
 this.*sleep*(10);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 }  
 else  
 {  
 synchronized (this) {  
 try {  
 this.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 }  
 public void pauseAI()  
 {  
 working=false;  
 }  
 public void resumeAI()  
 {  
 this.notify();  
 working = true;  
 }  
}

public class WoodAI extends BaseAI  
{  
 Habitat h;  
 WoodAI(Habitat habitat)  
 {  
 h=habitat;  
 }  
 public void run() {  
 double speed = 1000;  
 //System.out.println(h.houseSingleton.houseList.get(0).x);  
 //System.out.println(h.houseSingleton.houseList.get(0).y);  
 while (true) {  
 if (working) {  
 for (int i = 0; i < h.houseSingleton.houseList.size() - 1; i++) {  
  
 if (h.houseSingleton.houseList.get(i) instanceof Wood) {  
 double x1 = h.houseSingleton.houseList.get(i).x;  
 double y1 = h.houseSingleton.houseList.get(i).y;  
 double x2 = (h.ui.simulation\_panel.getWidth() / 2) \* Math.*random*() + h.ui.simulation\_panel.getWidth() / 2;  
 double y2 = (h.ui.simulation\_panel.getHeight() / 2) \* Math.*random*() + h.ui.simulation\_panel.getHeight() / 2;  
  
 double a = x2 - x1;  
 double b = y2 - y1;  
 double c = a \* a + b \* b;  
 double vx = (speed \* a) / c;  
 h.houseSingleton.houseList.get(i).setVx(vx);//System.out.println(vx);  
 double vy = (speed \* b) / c;  
 h.houseSingleton.houseList.get(i).setVy(vy);//System.out.println(vy);  
 int tact = h.houseSingleton.houseList.get(i).getTact();  
 int lasttact = (int) (c / speed);  
 h.houseSingleton.houseList.get(i).setLasttact(lasttact);  
 //System.out.println(h.houseSingleton.houseList.get(i).getTact());  
 //System.out.println(h.houseSingleton.houseList.get(i).getLasttact());  
 if (h.houseSingleton.houseList.get(i).getTact() < h.houseSingleton.houseList.get(i).getLasttact()) {  
 tact++;  
 h.houseSingleton.houseList.get(i).setTact(tact);  
 x1 = x1 + h.houseSingleton.houseList.get(i).getVx();  
 y1 = y1 + h.houseSingleton.houseList.get(i).getVy();  
  
 h.houseSingleton.houseList.get(i).setX(x1);  
 h.houseSingleton.houseList.get(i).setY(y1);  
 }  
 }  
 }  
 try {  
 this.*sleep*(10);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else  
 {  
 synchronized (this)  
 {  
 try {  
 this.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 }  
 public void pauseAI()  
 {  
 working=false;  
 }  
 public synchronized void resumeAI()  
 {  
 working = true;  
 this.notify();  
 }  
}

**Вывод:**

Были изучены особенности реализации и работы потоков, управления приоритетами и синхронизации потоков.