Федеральное агентство по образованию

федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Ивановский государственный энергетический университет имени В.И. Ленина»

Кафедра высокопроизводительных вычислительных систем

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI

Методические указания

Иваново 2014

Составитель С.Г.СИДОРОВ

Редактор И.Ф.ЯСИНСКИЙ

Методические указания предназначены для использования в качестве справочного материала при выполнении студентами, изучающими объектно-ориентированное программирование в среде Delphi, курсовой работы. Рассматриваются вопросы объектно-ориентированного программирования, реализации классов и создания динамических объектов в среде Delphi на языке программирования Object Pascal.

Рецензент
кафедра высокопроизводительных вычислительных систем ФГБОУВПО «Ивановский государственный энергетический университет имени В.И.Ленина»

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI
Методические указания

Составитель СИДОРОВ Сергей Георгиевич

Редактор Н.Б.Михалева

Подписано в печать Формат 60x84 1/16
Печать плоская. Усл.печ.л. 0,93. Тираж 30 экз. Заказ №

ФГБОУВПО «Ивановский государственный энергетический университет имени В.И.Ленина»
Отпечатано в УИУНЛ ИГЭУ
153003 г.Иваново, ул.Рабфаковская, 34

Содержание

Постановка задачи

Задача – разработать программу моделирования агентного взаимодействия с использованием объектно-ориентированного подхода.

Агент визуально представляет собой окружность заданного размера и цвета, изначально расположенную в заданных координатах и осуществляющую движение в заданном направлении. Начальное количество агентов – 100. Каждый из агентов имеет свой цвет и направление движения.

При столкновении агентов происходит их слияние. Агент, имеющий больший размер, поглощает агента меньшего размера. При слиянии увеличивается размер агента "хищника", изменяется его цвет в зависимости от цвета агента "жертвы", а также изменяется направление движения агента "хищника". Агент "жертва" уничтожается.

Моделирование ведется до момента невозможности дальнейшего сокращения популяции агентов (в идеале до одного агента).

Интерфейс программы

Интерфейс программы представлен на рис.1. Для формирования интерфейса программы использованы следующие компоненты (стартовое свойство Name компонентов изменено):

- FormMain (класс TForm) – главная форма;
- ToolBar (класс TToolBar) – панель инструментов;
- tbStartStop (класс TToolButton) – кнопка запуска программы;
- ToolButton1 (класс TToolButton) – разделитель в панели;
- Timer (класс TTimer) – таймер;
- ImageList (класс TImageList) – список битмапов кнопок;
- ActionList (класс TActionList) – список действий;
- ImageBox (класс TGroupBox) – фрейм для игрового поля;
- Image (класс TImage) – объект показа игрового поля;
- Status (класс TStatusBar) – статус-строка.

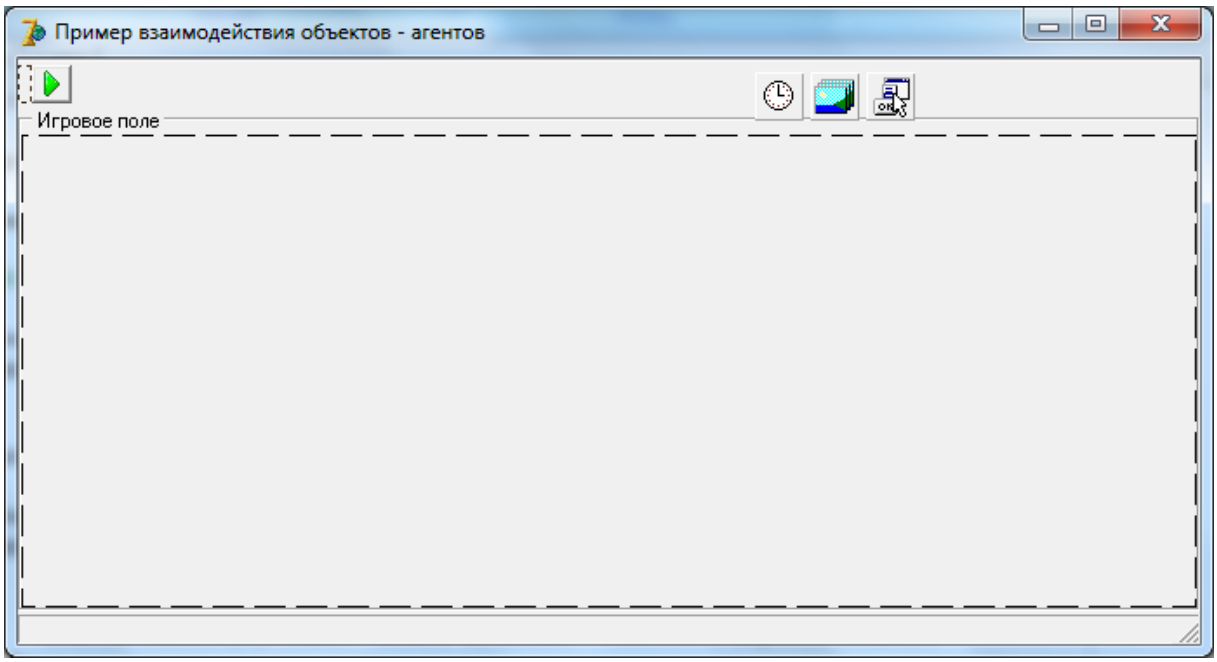

Рис. 1. Интерфейс программы

Свойства ряда установленных компонентов были изменены. Так, у компонента FormMain изменены свойства Caption на "Пример взаимодействия объектов–агентов" и свойство WindowState на wsMaximized (при старте программы форма разворачивается на полный экран). Свойство Enabled компонента Timer установлено в False (таймер по умолчанию выключен), а свойство Interval в 200 (интервал срабатывания таймера в милисекундах, т.е. 5 раз в секунду). В компонет Imagelist загружены пиктограммы для отображения на кнопке tbStartStop (рис.2).

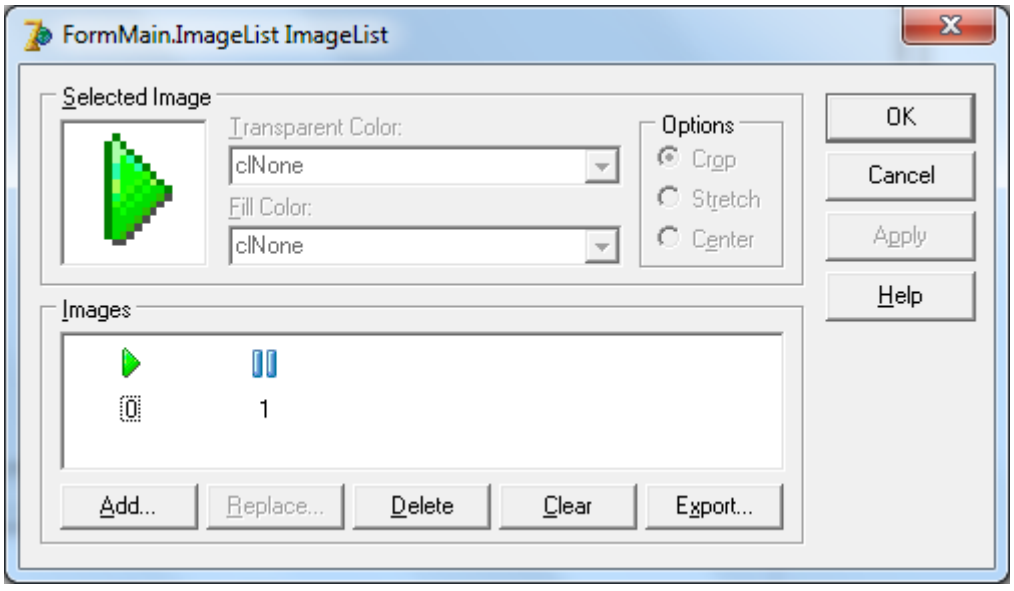

Рис. 2. Пиктограммы в ImageList

Для объекта ActionList создано действие acStartStop (рис.3) и свойству Images установлено значение ImageList (пиктограммы действиям будут браться из списка ImageList).

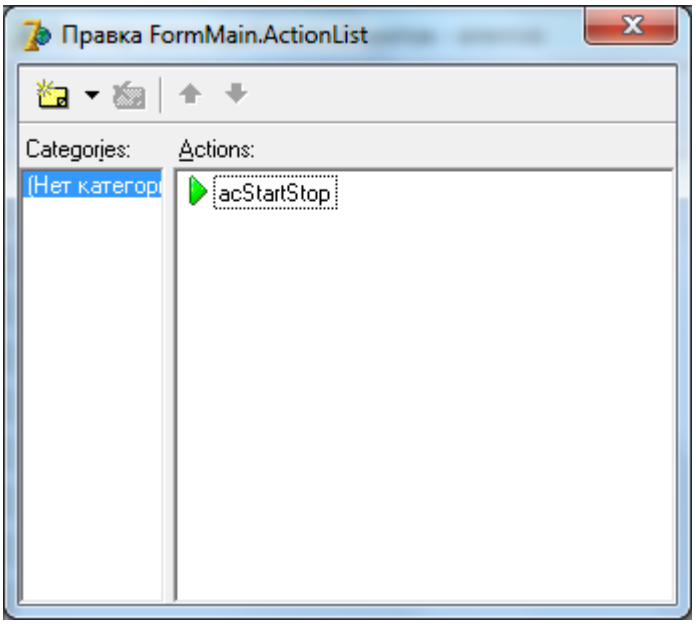

Рис. 3. Список действий ActionList

Свойство Align компонента ImageBox установлено в alClient (развернуть фрейм на всю форму). Свойство Align компонента Image установлено в alClient (игровое поле по размеру клиентской области фрейма). Для компонента Status создана панель вывода (рис.4).

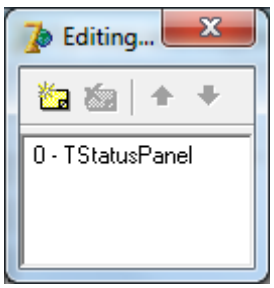

Рис. 4. Список панелей компонента Status

Компонент ToolBar также связан со списком пиктограмм. Его свойство Images установлено в ImageList. На компоненте ToolBar созданы кнопки ToolButton1 в качестве разделителя и кнопка tbStartStop (свойство Name изменено). Для кнопки tbStartStop свойству Action назначено действие acStartStop.

Код модуля, реализующего классы

```
////////////////////////////////////////////////////////////
//            Описание классов объектов - агентов         //
//               (c) Сидоров С.Г., ИГЭУ, 2013             //
//                 e-mail: sgsidorov@mail.ru              //
////////////////////////////////////////////////////////////

unit UnitObj;

interface

uses Graphics,Windows,ExtCtrls;

type
     TAgent = class   // описание класса агента
     private
       FRect :TRect;                    // координаты агента
       FCol  :TColor;                   // цвет
       FSize :longint;                  // размер
       FHX   :longint;                  // шаг по X
       FHY   :longint;                  // шаг по Y
       function GetX:longint;           // центр агента по X
       function GetY:longint;           // центр агента по Y
       function GetR:longint;           // радиус агента
       procedure SetR(R:longint);       // установка радиуса
       procedure SetHX(HX:longint);     // установка шага по X
       procedure SetHY(HY:longint);     // установка шага по Y
     public
       constructor Create;              // конструктор
       destructor Free;                 // деструктор
       procedure Move;                  // шаг агента
       procedure Show(Img:TImage);      // отрисовка агента
       property X  :longint read GetX;  // X центра
       property Y  :longint read GetY;  // Y центра
       property R  :longint read GetR write SetR;  // радиус
       property HX :longint read FHX  write SetHX; // шаг по X
       property HY :longint read FHY  write SetHY; // шаг по Y
       property Col:TColor  read FCol write FCol;  // цвет
     end;

     TAgents = class  // описание класса группы агентов
     private
       Items :array of TAgent;          // агенты
       function GetCount:longint;       // число агентов
       function MergeColor(A,B:TColor):TColor; // смесь цвета
       procedure Merge(A,B:longint);    // слияние агентов A,B
     public
       constructor Create;              // конструктор
       destructor Free;                 // деструктор
```

```
      function Add:TAgent;              // добавление агента
      procedure Del(N:longint);         // удаление агента N
      procedure Move;                   // шаг агента
      procedure Show(Img:TImage);       // отрисовка агентуры
      property Count :longint read GetCount; // число агентов
    end;

var  Agents :TAgents;   // агентура

implementation

var  Xmax :longint = 1000; // стартовая ширина экрана
     Ymax :longint = 600;  // стартовая высота экрана

////////////////////////////////////////////////////////////

constructor TAgent.Create; // конструктор (рождение агента)
begin
     FSize:=20;
     FRect.Left:=random(Xmax);  FRect.Right:=FRect.Left+FSize;
     FRect.Top:=random(Ymax);   FRect.Bottom:=FRect.Top+FSize;
     FCol:=RGB(random(255),random(255),random(255));
     FHX:=5-random(11);
     FHY:=5-random(11);
end;

destructor TAgent.Free; // деструктор (конец агента)
begin
end;

procedure TAgent.Move; // шаг агента
begin
     // шаг по горизонтали
     FRect.Left:=FRect.Left+FHX;
     FRect.Right:=FRect.Left+FSize;
     if (FRect.Left<0) or (FRect.Right>Xmax) then FHX:=-FHX;
     // шаг по вертикали
     FRect.Top:=FRect.Top+FHY;
     FRect.Bottom:=FRect.Top+FSize;
     if (FRect.Top<0) or (FRect.Bottom>Ymax) then FHY:=-FHY;
end;

procedure TAgent.Show(Img:TImage); // отрисовка агента
begin
     Img.Canvas.Pen.Color:=clBlue;
     Img.Canvas.Brush.Color:=FCol;
     Img.Canvas.Ellipse(FRect);
end;

function TAgent.GetX:longint;  // центр агента по X
begin
     result:=(FRect.Left+FRect.Right) div 2;
```

```
end;

function TAgent.GetY:longint;  // центр агента по Y
begin
     result:=(FRect.Top+FRect.Bottom) div 2;
end;

function TAgent.GetR:longint;  // радиус агента
begin
     result:=FSize div 2;
end;

procedure TAgent.SetR(R:longint); // установка радиуса агента
begin
     FSize:=R*2;
end;

procedure TAgent.SetHX(HX:longint); // установка шага по X
begin
     if Abs(HX)<1 then FHX:=1-random(3) else FHX:=HX;
end;

procedure TAgent.SetHY(HY:longint); // установка шага по Y
begin
     if Abs(HY)<1 then FHY:=1-random(3) else FHY:=HY;
end;

////////////////////////////////////////////////////////////

constructor TAgents.Create; // конструктор (рождение агентуры)
var  i :longint; // индекс
begin
     for i:=1 to 100 do Add;
end;

destructor TAgents.Free; // деструктор (конец агентуры)
var  i :longint; // индекс
begin
     for i:=0 to Length(Items)-1 do Items[i].Free;
end;

function TAgents.Add:TAgent;  // добавление агента
var  L :longint; // размер массива агентов
begin
     L:=Length(Items);
     SetLength(Items,L+1);
     Items[L]:=TAgent.Create;
     result:=Items[L];
end;

procedure TAgents.Del(N:longint);  // удаление агента N
var  L :longint; // размер массива агентов
```

```
     i :longint; // индекс
begin
     L:=Length(Items); if (N<0)or(N>=L) then Exit;
     Items[N].Free;
     for i:=N to L-2 do Items[i]:=Items[i+1];
     SetLength(Items,L-1);
end;

function TAgents.MergeColor(A,B:TColor):TColor; // смешение
цветов
var  RA,GA,BA,RB,GB,BB :byte;  // оттенки
begin
     RA:=GetRValue(A);  GA:=GetGValue(A);  BA:=GetBValue(A);
     RB:=GetRValue(B);  GB:=GetGValue(B);  BB:=GetBValue(B);
     result:=RGB((RA+RB) div 2, (GA+GB) div 2, (BA+BB) div 2);
end;

procedure TAgents.Merge(A,B:longint); // слияние агентов A и B
var  V :longint;
begin
     // выбор хищника A и жертвы B
     if Items[A].FSize<Items[B].FSize then begin
        V:=A; A:=B; B:=V;
     end;
     // слияние
     Items[A].R:=Items[A].R+Items[B].R div 3;
     Items[A].Col:=MergeColor(Items[A].Col,Items[B].Col);
     Items[A].HX:=(Items[A].HX+Items[B].HX) div 2;
     Items[A].HY:=(Items[A].HY+Items[B].HY) div 2;
     Del(B);
end;

procedure TAgents.Move; // шаги агентуры
var  i,j :longint; // индексы
     R   :double;  // расстояния между агентами
begin
     // смещения агентов
     for i:=0 to Length(Items)-1 do Items[i].Move;
     // слияние агентов
     i:=0;
     While i<Length(Items)-1 do begin
           j:=i+1;
           While j<Length(Items) do begin
                 R:=sqrt(sqr(Items[i].X-Items[j].X)
                        +sqr(Items[i].Y-Items[j].Y));
                 if (R<Items[j].R)or(R<Items[i].R)
                    then Merge(i,j);
                 j:=j+1;
           end;
           i:=i+1;
     end;
end;
```

```
procedure TAgents.Show(Img:TImage); // отрисовка агентуры
var  i :longint; // индекс
begin
     // фиксация изменения размеров поля
     Xmax:=Img.Picture.Width-1;
     Ymax:=Img.Picture.Height-1;
     // очистка поля
     Img.Canvas.Brush.Color:=clWhite;
     Img.Canvas.FillRect(Img.ClientRect);
     // отрисовка агентов
     for i:=0 to Length(Items)-1 do Items[i].Show(Img);
end;

function TAgents.GetCount:longint;  // число агентов
begin
     result:=Length(Items);
end;

initialization
    Agents:=TAgents.Create;
finalization
    Agents.Free;
end.
```

Код модуля формы

```
unit UnitMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, ToolWin, ComCtrls, ActnList,
ImgList;

type
  TFormMain = class(TForm)
    ImageBox: TGroupBox;
    Image: TImage;
    ToolBar: TToolBar;
    Status: TStatusBar;
    Timer: TTimer;
    ToolButton1: TToolButton;
    tbStartStop: TToolButton;
    ImageList: TImageList;
    ActionList: TActionList;
    acStartStop: TAction;
    procedure FormActivate(Sender: TObject);
    procedure TimerTimer(Sender: TObject);
    procedure acStartStopExecute(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormMain: TFormMain;

implementation

uses UnitObj;

{$R *.dfm}

procedure TFormMain.FormActivate(Sender: TObject); //активация
begin
     Agents.Show(Image);
     Status.Panels[0].Text:='Всего агентов:
'+IntToStr(Agents.Count);
end;

procedure TFormMain.TimerTimer(Sender: TObject); // таймер
```

```
begin
     Agents.Move;
     Agents.Show(Image);
     Status.Panels[0].Text:='Всего агентов:
'+IntToStr(Agents.Count);
end;

procedure TFormMain.acStartStopExecute(Sender: TObject);
begin
     Timer.Enabled:=not Timer.Enabled;
     if Timer.Enabled then acStartStop.ImageIndex:=1
                      else acStartStop.ImageIndex:=0;
end;

end.
```

Описание программы

В модуле UnitObj приведены описания двух классов:

- TAgent – описывающий поведение агента;
- TAgents – описывающий поведение коллекции агентов.

Доступ к полям класса агента осуществляется через его свойства X,Y – координаты центра, R – радиус, HX,HY – смещения по осям X и Y, Col – цвет агента. Свойства агента размещены в открытых объявлениях и доступны извне модуля, поля агента скрыты и недоступны извне модуля.

Каждое из свойств содержит описание типа, указание на поле или функцию, используемую при чтении значения свойства, поле или процедуру, используемую при записи значения в свойство.

Инициализация полей агента производится в конструкторе, при создании экземпляра класса. При этом используются значения глобальных переменных Xmax и Ymax, определяющих максимальные значения размера игрового поля. Изначально они заданы константами, но в процессе прорисовки коллекции классов производится их уточнение по реальным размерам объекта Image, на котором отображаются агенты.

Метод Move осуществляет перемещение агента и в случае достижения границы игрового поля меняет направление движения агента по соответствующей оси на противоположное.

Метод Show вызывает прорисовку агента на объекте Image цветом Col.

Остальные функции класса агента очевидны и в отдельном описании не нуждаются.

Класс TAgents содержит коллекцию агентов – объектов класса TAgent, описываемую динамическим массивом Items. Добавление

нового агента в коллекцию производится методом Add. Удаление агента из коллекции производится методом Del с указанием номера удаляемого агента. Инициализация коллекции производится в конструкторе Create класса.

Метод Move вызывает перемещение агентов, входящих в коллекцию, после чего производится попарный анализ их пересечения. В случае сближения пары агентов на расстояние, меньшее радиуса одного из агентов, вызывается метод Merge, производящий выбор "хищника" и "жертвы", а также осуществляющий поглощение агента "жертвы" агентом "хищником".

В процессе попарного сравнения применены операторы цикла While, а не For, как это видится на первый взгляд. Это сделано специально, чтобы избежать проблем при сокращении размера коллекции и возможной некорректной работы операторов цикла For.

Метод Show производит очистку игрового поля и производит отрисовку агентов, входящих в коллекцию. При этом уточняются максимальные размеры игрового поля по реальным размерам объекта Image.

Создание объекта Agents класса TAgents реализовано непосредственно в модуле UnitObj. Это сделано для упрощения отслеживания процессов создания и удаления данного экземпляра класса TAgents. Соответствующие вызовы конструктора и деструктора размещены в разделах инициализации и финализации соответственно.

Для отображения процесса моделирования в режиме реального времени в главном модуле UnitMain создан обработчик таймера TimerTimer. В его коде вызываются методы Move и Show коллекции агентов Agents класса TAgents и, в статус-строку выводится количество агентов, входящих в коллекцию (свойство Count коллекции Agents).

Запуск и остановка работы таймера осуществляется с помощью действия acStartStopExecute, назначенного обработчиком кнопки tbStartStop панели инструментов. В обработчике производится поочередное изменение свойства Enabled таймера, отвечающего за работу таймера. Также в этой процедуре изменяется пиктограмма на кнопке панели инструментов, отображающая текущее состояние процесса моделирования. Выбор пиктограммы осуществляется изменением индекса изображения в объекте – действии acStartStop.

Начальная инициализация игрового поля и статус-строки производится в обработчике активации формы FormActivate.

Пример работы программы

В начальный момент работы программы (рис.5) на игровом поле расположены 100 агентов в виде кружочков небольшого размера, разбросанных в случайном порядке по всей площади и имеющих случайную цветовую окраску. Моделирование не запущено (кнопка на панели инструментов имеет вид зеленой стрелки). В статус-строке отображается количество агентов на игровом поле – 100.

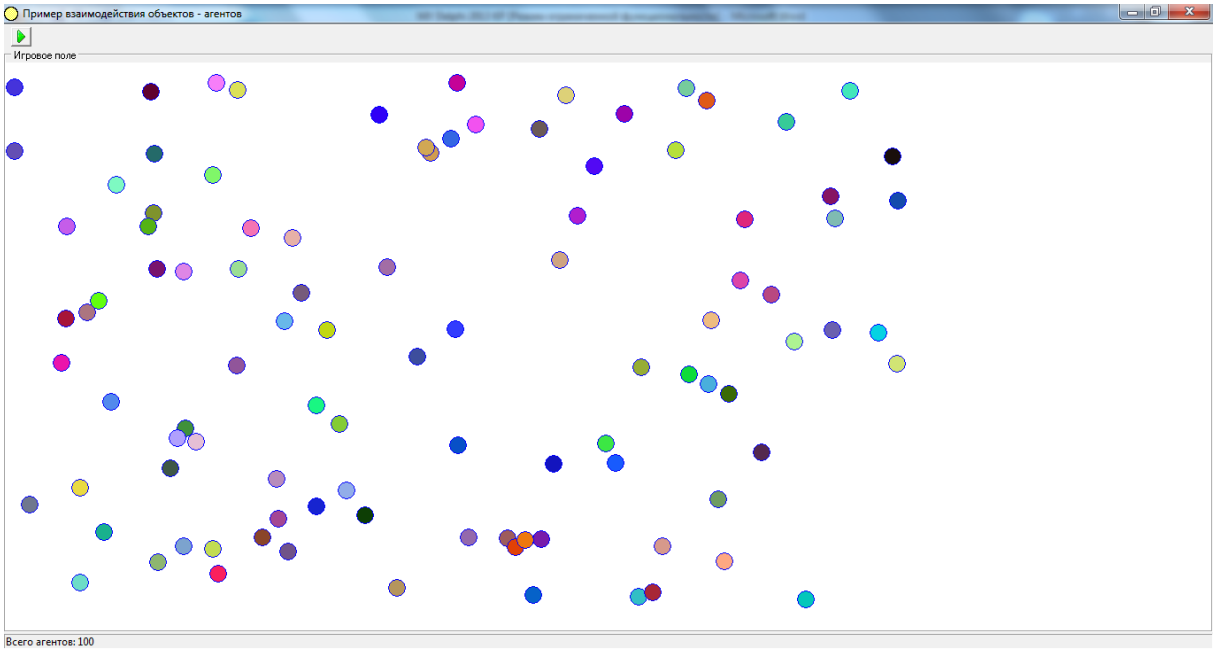

Рис. 5. Начальная стадия работы программы

Для запуска процесса моделирования используется кнопка на панели инструментов. После ее нажатия пиктограмма на ней меняется на синее изображение паузы. Поочередное нажатие на эту кнопку будет вызывать остановку процесса моделирования и его продолжение.

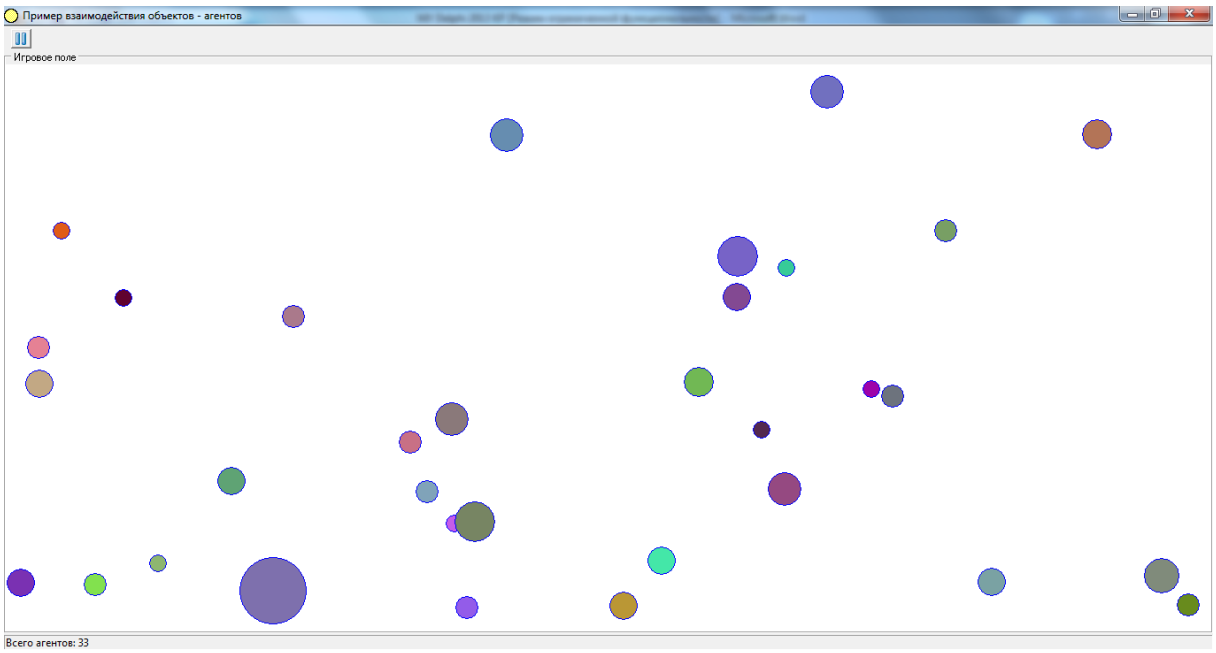

Рис. 6. Текущая стадия работы программы

Спустя несколько десятков секунд моделирования картина на игровом поле меняется (рис.6). Агентов становится меньше. Размеры агентов также начинают отличаться. Агенты "хищники", поглотившие агентов "жертв", становятся крупнее.

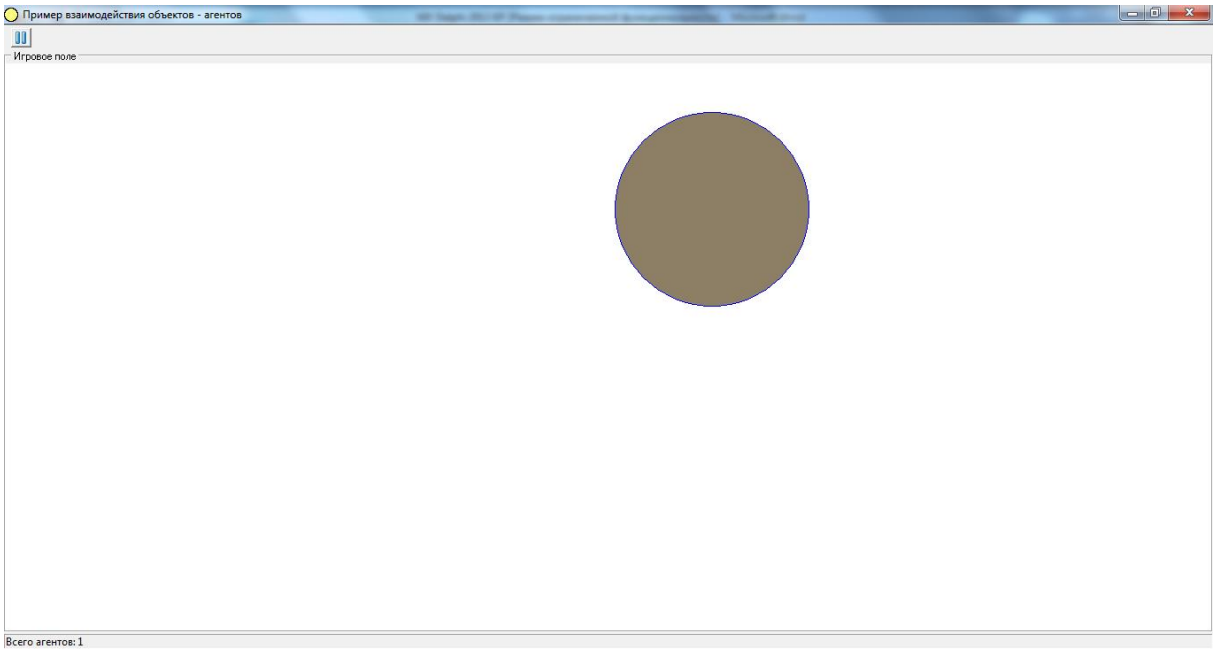

Рис. 7. Финальная стадия работы программы

Примерно через 10-15 минут работы программы все агенты поглощены и на игровом поле остается последний агент с максимальным размером и совокупной серой цветовой окраской.

В силу случайности процесса разброса агентов по игровому полю и различных стартовых траекторий движения итоговый результат может не совпадать с представленным на рис.7.

Так, в некоторых ситуациях студентами группы 2-46 были получены финальные результаты с несколькими агентами, траектории движения которых не пересекались.

Примерные темы курсовых работ

Вариант темы курсовой работы либо собственная тема выбирается по согласованию с преподавателем.

При выполнении курсовой работы необходимо создать классы, реализующие функции в соответствии с темой. Написать программу, реализующую разработанные классы.

№	Тема
1	Агентное моделирование движения транспорта с анализом эффективности средств управления транспортными потоками
2	Оценка производительности канала связи и передача произвольных данных через сеть в режиме реального времени
3	Разработка компьютерной игры с элементами агентного моделирования
4	Построение системы распределенных вычислений в сети на основе сценариев, автоматизации распараллеливания и сбора результатов
5	Построение динамических 3D моделей для отображения векторных и скалярных полей с использованием OpenGL
6	Разработка классов реализации быстрого преобразования Фурье для анализа статических и динамически меняющихся рядов данных
7	Создание классов предобработки изображений на основе цифровой фильтрации для поиска и выделения признаковой информации в системах распознавания
8	Разработка мультстудии на основе объектов с автоматизированным расчетом траекторий перемещений, морфирования изображений и цветов
9	Реализация классов асиммитричного шифрования файлов и реализации электронной цифровой подписи
10	Построение дизассемблера машинного кода и автоматический анализ структуры исследуемого файла
11	Оптимизация кода программ на языке высокого уровня (Си, Паскаль, Фортран, Бэйсик)
12	Конвертор кода программ на языках высокого уровня (Си, Паскаль, Фортран, Бэйсик)
13	Разработка двоичного и текстового редакторов произвольных файлов с отображением кода в различных системах кодирования

14	Построение системы предпечатной обработки сборных текстовых документов в формате Word, использующих различные шаблоны и стили
15	Разработка программы многопользовательской сетевой коммуникации на основе передачи текстов
16	Разработка программы многопользовательской сетевой коммуникации на основе передачи звуков
17	Разработка программы многопользовательской сетевой коммуникации на основе передачи видеоизображений
18	Создание компонентов видеозахвата изображений с web-камеры в режиме реального времени
19	Создание компонентов аудиозахвата звукового ряда с микрофона в режиме реального времени
20	Построение системы анализа сложных программных комплексов на основе графического представления структуры проекта
21	Разработка компонентов чтения и обработки спутниковых карт из общедоступных источников maps.google.com, maps.yandex.ru
22	Создание компонентов, реализующих элементы искусственных нейронных сетей, и разработка редактора сборки компонентов в нейросетевую систему
23	Создание компонентов, реализующих элементы генетического алгоритма, и разработка редактора сборки компонентов в систему генетического поиска
24	Разработка программы планировщика расписаний на основе многокритериальной оптимизации
25	Разработка компонентов, реализующих арифметические и логические операции обработки "больших" чисел для применения в системах криптографической защиты
26	Решение систем линейных уравнений различными методами (Гаусса, подстановки, формулам Крамера и т.д.)
27	Решение систем нелинейных уравнений различными методами (Ньютона, Зейделя и т.д.)
28	Создание компонентов, реализующих различные алгоритмы сжатия цветных и монохромных изображений (RLE, JPEG, DJVU, GIF)
29	Создание компонентов, реализующих различные алгоритмы сжатия моно- и стереосигналов звукового ряда (WAV, MP3)
30	Разработка редактора построения логических схем из отдельных логических элементов и моделирование работы построенной схемы

Контрольные вопросы

1. Опишите структуру класса.
2. Как создать объект заданного класса?
3. Чем свойство объекта отличается от поля?
4. Чем метод объекта отличается от обычных подпрограмм?
5. Как задать область видимости полей и методов?
6. Что такое конструктор и как его использовать?
7. Что такое деструктор и как его использовать?
8. Как создать объект на основе существующего класса?
9. Как переопределить методы объекта-предка?
10. Что такое виртуальные методы?
11. Как вызвать код метода предка в методе наследнике?
12. Как задать значения свойств по умолчанию?