УДК 004.92

РАЗРАБОТКА И РЕАЛИЗАЦИЯ БАЗОВОГО АЛГОРИТМА ФРАКТАЛЬНОГО КОДИРОВАНИЯ ИЗОБРАЖЕНИЙ

А.В. Савкина
Мордовский государственный университет им. Н. П. Огарева

Аннотация. В качестве базового алгоритма фрактального кодирования изображений предложен алгоритм с использованием разбиения изображения на неперекрывающиеся ранговые блоки, приведена блок-схема, в которой представлены основные шаги фрактального кодирования изображений, приведены отдельные фрагменты кода базового алгоритма, рабочие и выходные формы. Реализованный алгоритм выполняет фрактальное кодирование изображения перед помещением его образа в виде квадродерева в базу данных с последующим анализом в процессе осуществления поиска.
В реализации разработанного базового алгоритма используется система Microsoft Visual Studio .NET, которая представляет удобную среду разработки для создания Windows и .NET приложений, Web-приложений и XML Web-сервисов.
Поиск изображения с помощью предложенного базового алгоритма фрактального кодирования осуществляется по негативу, цветному и черно-белому изображению.

Ключевые понятия: фрактальное кодирование, хранение изображения, поиск по образцу, база данных, алгоритм.

Любая информационная система предназначена для сбора, хранения и обработки информации, поэтому в основе любой информационной системы лежит среда хранения и доступа к данным. Она должна обеспечивать уровень надежности хранения и эффективность доступа, которые соответствуют области применения информационной системы. Основным требованием к системе, использующей математический аппарат, является ее быстродействие. Разработанный комплекс программ позволяет решать поставленную задачу с помощью фрактального кодирования.

Большие коллекции цифровых изображений появляются в современной жизни довольно часто. Некоторые коллекции цифровых изображений оцифровываются в надежде лучшей сохранности, более легкого распространения и лучшего доступа при наличии современного оборудования и новых информационных технологий. Другие являются цифровыми изначально; это, например, личные коллекции семейных

фотографий (которые могут быть большими и цифровыми), коллекции домашнего видео (также могут иметь значительный размер и многие из них в настоящее время являются цифровыми), кроме того, большая, неорганизованная коллекция – это Web.

Существующие средства взаимодействия с коллекциями документов или данных довольно сложны. Как правило, для поиска в коллекции используются различные разновидности сравнения с текстовым шаблоном, кластеризация коллекций текста или же методы информационной проходки (data mining). Информационная проходка включает использование сложных процедур статистического подбора для поиска ранее неизвестных трендов; это полезное занятие известно под именем «пояснительный анализ данных» (exploratory data analysis) или «добыча данных» (data dredging). Обычно ценность коллекции во многом определяется наличием подобного средства.

В настоящее время удовлетворительно организовать коллекцию изображений или провести поиск в ней сложно, т.е. эти коллекции в чем-то сродни плохо организованному книжному магазину. Сложность заключается в построении подходящего представления информации, содержащейся в изображении. Вручную аннотировать каждую картину нет смысла, поскольку подготовка хорошего текстового описания изображения — это весьма сложная задача. Более того, некоторые коллекции огромны. Ручная индексация большой коллекции требует значительного объема работы, а в перспективе может потребоваться переиндексация разделов коллекции (например, вследствие какого-то события ранее неизвестный человек стал знаменитым и требуется узнать, содержатся ли его изображения в коллекции). Наконец, часто просто трудно сказать, о чем данная картина.

Несмотря на все эти трудности, любая технология, которая помогает управлять коллекциями изображений, имеет множество сфер практического применения. Одним из важных средств является поиск («найти изображение, отвечающее таким критериям»), но это не единственное требуемое средство. Например, изображения нужно организовать так, чтобы была возможной навигация: изображения со сходным содержимым должны располагаться рядом. Или же требуется организовать поиск трендов, или нужно средство, фиксирующее важные изменения.

Изображение с помощью предложенной программной разработки находится корректно, если образец отличается от искомого некоторыми свойствами, такие как: размер, яркость, контрастность, небольшая зашумленность, незначительное изменение цветов, размытие, поиск по негативу, поиск осуществляется и по черно-белому изображению.

В реализации комплекса программ используется теория, представленная в предыдущих отчетах и Microsoft Visual C# из пакета Microsoft Visual Studio.NET.

Система Visual Studio.NET не только обеспечивает возможность создания приложений и веб-служб XML, доступными для использования лю-

быми обозреватель и практически любым устройством, но эта система также позволяет применять для разработки любой язык программирования.

Microsoft Visual Studio .NET Enterprise Architect используется для решения следующих задач:

− создания эффективных шаблонов в области построения архитектуры приложений и обеспечения доступа к ним для всех участников группы разработки; формирования и распространения четких инструкций по разработке программ для обеспечения совместного использования рекомендаций в среде Visual Studio .NET с помощью языка описания шаблонов (Template Description Language) и корпоративных шаблонов проектов; упрощения разработки сложных приложений с использованием корпоративных схем;

− наглядного моделирования Web-служб XML и баз данных с помощью средств Microsoft Visio; наглядного описания архитектуры программного обеспечения и распространения информации о ней; применения диаграмм для вариантов использования (use-case), классов (class) и действий (activity), согласно спецификации языка UML; быстрого проведения инженерного анализа и генерации структуры программного кода; полной поддержки концептуального, логического и физического моделирования баз данных, обеспечения точности формулировки бизнес-требований и четкой постановки задачи перед разработчиками;

− постоянной реализации преимуществ использования современного средства разработки, занимающего лидирующие позиции в своей области; быстрого создания и тестирования XML Web-служб и приложений; применения системы Visual Studio .NET для быстрой разработки прототипов, оценки альтернативных вариантов и создания многократно используемых компонентов.

Из пакета Microsoft Visual Studio .NET Enterprise Architect используется средство Microsoft Visual C# .NET, которое предоставляет удобную среду разработки для создания Windows-приложений и .NET-приложений, Web-приложений и XML Web-сервисов. Visual C# .NET включает ставшую стандартом отрасли библиотеки ATL (Active Template Library) и MFC (Microsoft Foundation Class), расширения языка C#, а также мощную интегрированную среду разработки (IDE), позволяющую разработчикам эффективно создавать и отлаживать код.

В сочетании с .NET Common Language Runtime C# может применяться для разработки объектно-ориентированных программ с привлечением всего опыта C, C++ или COM, что позволит создавать быстрые и мощные приложения. Поддерживая шаблоны и имея оптимизирующий компилятор, Visual C# .NET генерирует высокоэффективные приложения и компоненты. Данный продукт позволяет разрабатывать целый комплекс Windows-приложений, включая Web-приложения, приложения для мобильных устройств, а также решения для «тонкого клиента»[1].

При фрактальном кодировании изображений производится поиск множества сжимающих преобразований, которые отображают доменные

блоки (которые могут перекрываться) в множество ранговых блоков, которые покрывают изображение. Ранговые блоки могут быть одинакового размера, но удобнее используется адаптивное разбиение с переменным размером блоков. Ранговые блоки, показанные на рисунке справа, являются результатом разбиения методом квадродерева.

В качестве базового алгоритма фрактального кодирования изображений выберем следующий:

1. Разбиваем изображение f на неперекрывающиеся ранговые блоки $\{R_i\}$. Для упрощения реализации алгоритма в качестве ранговых блоков возьмем прямоугольники. Блоки R_i могут быть равными, но чаще используется адаптивное разбиение с переменным размером блоков. Это дает возможность плотно заполнять ранговыми блоками маленького размера части изображения, содержащие мелкие детали.
2. Покрываем изображение последовательностью доменных блоков, возможно перекрывающихся. Домены могут быть разных размеров, и обычно их количество исчисляется сотнями и тысячами. Ниже обсуждаются схемы построения множества доменных блоков.
3. Для каждого рангового блока находим домен и соответствующее преобразование, которое наилучшим образом покрывает ранговый блок. Обычно это аффинное преобразование вида $w_i(f)(x, y)$. Настраиваем параметры преобразования, такие как контрастность и яркость, для наилучшего соответствия.
4. Если достаточно точного соответствия не получилось, то разбиваем ранговые блоки на меньшие ранговые блоки. Продолжаем этот процесс до тех пор, пока или не добьемся приемлемого соответствия, или размер ранговых блоков не достигнет некоторого заранее определенного предела [2].

При подгонке доменных блоков к ранговым блокам алгоритм реализует пространственную составляющую аффинных преобразований, включающую операции параллельного переноса, поворота и сжатия. Сжатие уменьшает размер доменного блока до размеров рангового блока. Эта операция дополняется простым усреднением по строкам и столбцам.

Шаг 3 требует наибольших вычислений. Для каждого рангового блока R_i алгоритм ищет домен D_i пространственное преобразование w'_i контрастность s_i и яркость o_i, такие, чтобы $w_i(f)$ было близко к изображению f в блоке R_i. То есть мы ищем w_i такое, чтобы величина

$$\int_{R_i} |w_i(f)(x,y) - f(x,y)|^2 \, dx \, dy \qquad (1)$$

была небольшой. Для оцифрованных изображений этот интеграл заменяется суммированием по пикселам. Если после нахождения наилучшего w_i, величина оказывается все еще больше некоторой заранее определенной погрешности, то адаптивная схема разбиения на блоки разбивает ранговые блоки на меньшие ранговые блоки и затем поиск оптимального преобразования повторяется для этих меньших блоков. Этот процесс продолжается до тех пор, пока величина интеграла не станет меньше

допустимой погрешности или пока не достигается заранее определенный минимальный размер рангового блока [3].

«Код» закодированного фрактальными методами изображения – это список, содержащий сведения о каждом ранговом блоке. А именно: расположение рангового блока, домен (обычно определяемый при помощи индекса), который отображается в этот ранговый блок, и параметры, описывающие преобразование доменного блока в ранговый. Таким образом, коэффициент сжатия зависит от количества ранговых блоков, а также от эффективности хранения информации о каждом ранговом блоке. Большое количество ранговых блоков обеспечивает высокое качество декодированного изображения, но за счет степени сжатия.

В программе для разбиения изображения на ранговые блоки используется функция FraktalCoding:

```
     public SCod FraktalCoding(int[,] Rangs,int minx,int miny,int
maxx,int maxy,int[,,,] Domens,int glub,SCod Cod,Graphics gr,Pen P)
     {
     int[,] temp=new int[maxx-minx,maxy-miny];
     int xd=0,yd=0;
     Contrast Cont,BestCont;
     int BestDomen=0,BestModif=0;
     double Eps,MinEps=double.MaxValue;
     BestCont.o=0;
     BestCont.s=0;
     if             (((maxx-minx)>=Consts.WidthDomens)||((maxy-miny)>=
Consts.HeightDomens))
     {
     if (glub<=Consts.maxglubina)
     {
     Cod.ArrayCod[Cod.NumLine,glub]=1;
     if                                    (F4.checkBox1.Checked==true)
gr.DrawRectangle(P,minx,miny,(maxx-minx)/2,(maxy-miny)/2);
     Cod=FraktalCoding(Rangs,minx,miny,(maxx+minx)/2,(maxy+miny)/2,Dom
ens,glub+1,Cod,gr,P);
     Cod.ArrayCod[Cod.NumLine,glub]=2;
     ...
     Cod.ArrayCod[Cod.NumLine,glub]=3;
     ...
     Cod.ArrayCod[Cod.NumLine,glub]=4;
     ...
     Cod=CopyArray(Cod,Cod.NumLine,glub);
     glub--;
     return Cod;
     }
     }

     for (yd=0;yd<Consts.NumDomensY;yd++)
     {
     if (MinEps==0) break;
     for (xd=0;xd<Consts.NumDomensX;xd++)
     {
     temp=ResizeArray(Domens,xd,yd,maxx-minx);
     Cont=SearchContrast(temp,Rangs,minx,miny,maxx,maxy);
     temp=ApplyContrast(temp,maxx-minx,maxx-minx,Cont);
     Eps=SearchEps(temp,Rangs,minx,miny,maxx,maxy);
```

```
      if (Eps<MinEps)
      {
      MinEps=Eps;
      BestDomen=xd+yd*Consts.NumDomensX;
      BestModif=0;
      BestCont=Cont;
      if (MinEps==0) break;
      }
      temp=Rotate90(temp,maxx-minx);
      Eps=SearchEps(temp,Rangs,minx,miny,maxx,maxy);
      if (Eps<MinEps)
      {
      ...
      }
      ...
      }
      }

      if ((MinEps<Consts.Eps)||(glub>Consts.maxglubina))
      {
      Cod=AddZero(Cod,glub);
      Cod.ArrayCod[Cod.NumLine,8]=(float) BestDomen;
      Cod.ArrayCod[Cod.NumLine,9]=(float) BestModif;
      Cod.ArrayCod[Cod.NumLine,10]=(float) BestCont.o;
      Cod.ArrayCod[Cod.NumLine,11]=(float) BestCont.s;
      glub--;
      Cod=CopyArray(Cod,Cod.NumLine,glub);
      Cod.NumLine++;
      return Cod;
      }

      else//if (glub<=Consts.maxglubina)
      {
      Cod.ArrayCod[Cod.NumLine,glub]=1;
      if                                    (F4.checkBox1.Checked==true)
gr.DrawRectangle(P,minx,miny,(maxx-minx)/2,(maxy-miny)/2);
      Cod=FraktalCoding(Rangs,minx,miny,(maxx+minx)/2,(maxy+miny)/2,Dom
ens,glub+1,Cod,gr,P);
      Cod.ArrayCod[Cod.NumLine,glub]=2;
      ...
      Cod.ArrayCod[Cod.NumLine,glub]=3;
      ...
      Cod.ArrayCod[Cod.NumLine,glub]=4;
      ...
      }

      glub--;
      return Cod;
      }
```

Одними из основных параметров является Rangs и Domens, которые являются массивами, соответственно ранговых и доменных блоков. Возвращает данная функция структуру типа SCod, которая хранит в себе массив, состоящий из индексов ранговых блоков, доменных блоков и индекс соответствующих преобразований, а так же данная структура хранит количество элементов в массиве. При первом запуске, ранговый блок

представляет из себя все изображение. Далее он сравнивается с доменным, если ранговый блок больше либо равен доменному, то ранговый блок разбивается на четыре части и для каждой из них снова запускается функция FractalCodding и происходит запись в структуру SCod. После того как ранговый блок стал меньше доменного, функция начинает оттискивать наиболее подходящий для него доменный блок, путем его поворота, сжатия и оттискивания наиболее оптимальной яркости и контрастности. Сравнивая эти блоки, вычисляется погрешность. Если погрешность больше заданной и максимальная глубина квадродереве не достигнута, то ранговый блок снова делится на четыре части, и для каждой части алгоритм повторяется. Если максимальная глубина достигнута, то сохраняем наиболее подходящий ранговый блок.

Функцию FraktalCoding приходится вызывать трижды, для каждого из каналов палитры RGB.

На рис.1 представлена блок-схема процесса фрактального кодирования.

Разработанная база данных для хранения необходимой информации состоит из трех связных между собой таблиц (рис. 2). Первая таблица – «Категория изображения» (Categor) – для удобного просмотра и поиска изображений. Следующая сущность, связанная с предыдущей, это «Изображение» (Image). В ней будет храниться название изображения и непосредственно само изображение. Так же имеются фрактальные коэффициенты, которые служат для сравнивания изображений, поэтому для каждой картинки необходимо хранить ещё и коэффициенты. Их хранение будет осуществляться в отдельной, связанной с предыдущей сущностью – таблице «Фрактальные коэффициенты» (Fractals).

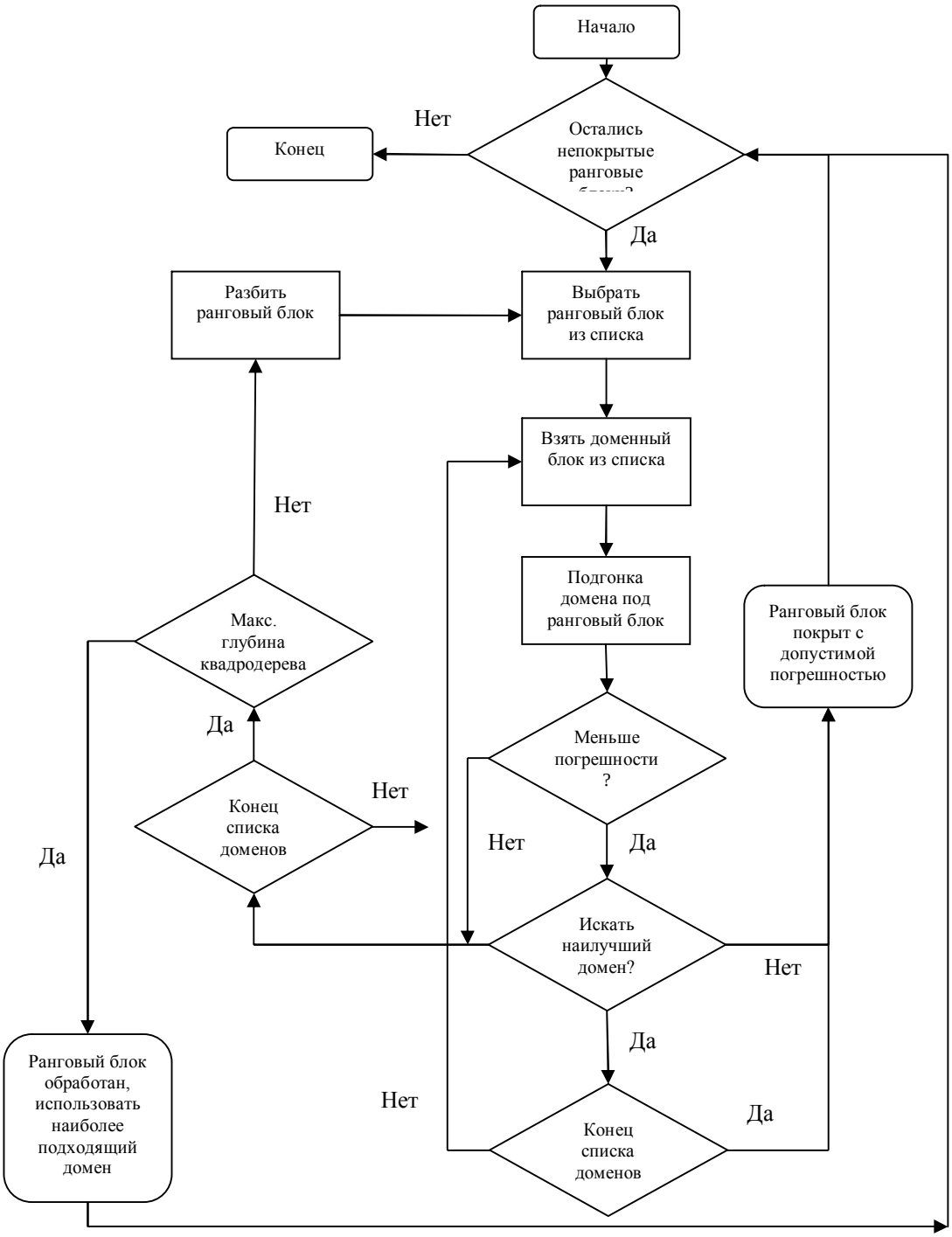

Рис. 1. Блок-схема показывает основные шаги фрактального кодирования изображений, реализованного в программе

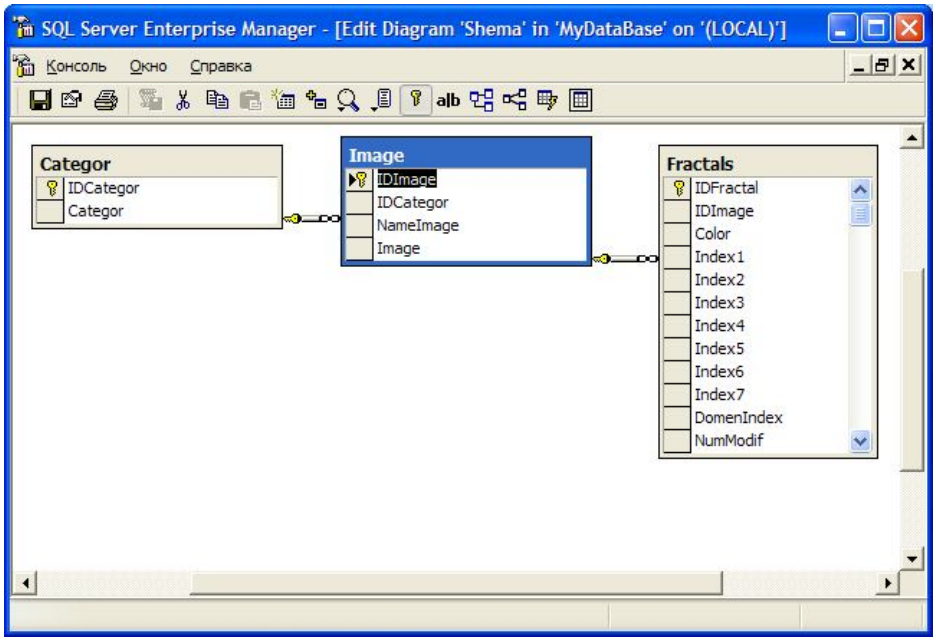

Рис. 2. Схема базы данных

На рисунке 3 представлена главная форма в режиме конструирования.

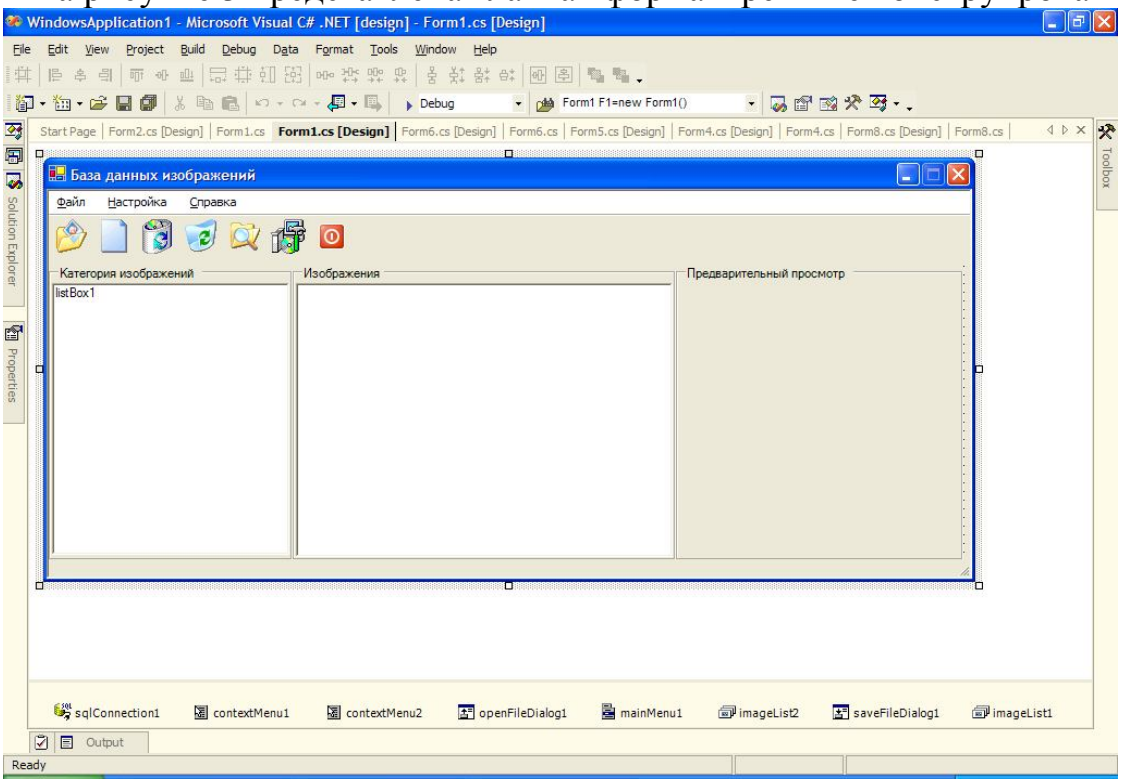

Рис. 3. Главная форма в режиме конструирования

Рабочие окна программы приведены на рис. 4-6.

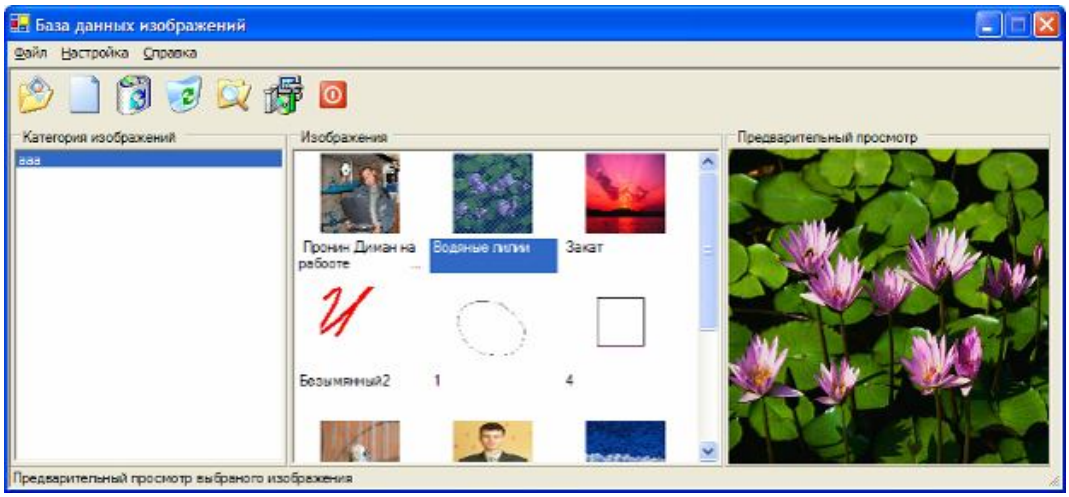

Рис. 4. Главная форма

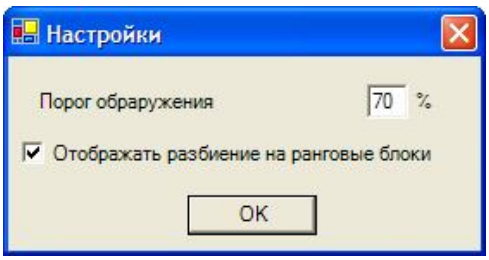

Рис. 5. Форма настроек

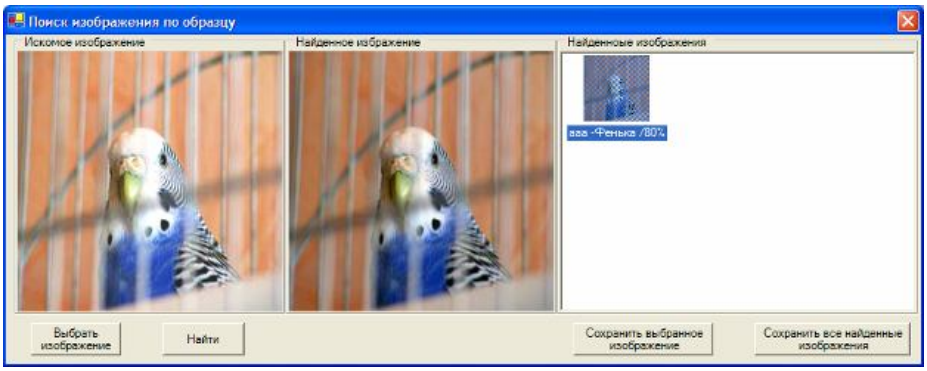

Рис. 6. Форма поиска изображений

Список использованной литературы

1. Платт, Д.С. Знакомство с Microsoft .NET / Пер. с англ./ Д.С. Платт. М.: Издательско-торговый дом "Русская редакция", 2001. С. 240.
2. Федер, Е. Фракталы / Пер. с англ./ Е. Федер. М.: Мир, 1991. С. 254.
3. Уэлстид, С. Фракталы и вейвлеты для сжатия изображения в действии. Учебное пособ. / С. Уэлстид. М.: Издательство Триумф, 2003. С. 320.

Сведения об авторе

Савкина Анастасия Владимировна, аспирантка Мордовского государственного университета им. Н.П.Огарева, преподаватель кафедры автоматизированных систем обработки информации и управления

Научный руководитель – Федосин Сергей Алексеевич, к.т.н., профессор, зав. кафедрой «Автоматизированные системы обработки информации и управления, 290-603, fedosinsa@mrsu.ru