РЕАЛИЗАЦИЯ ИНТЕРАКТИВНЫХ ЭЛЕМЕНТОВ ОБУЧАЮЩИХ ПРОГРАММ НА ЯЗЫКЕ СИ#

Ахмадеев В.А.
Волкова Т.И., к.п.н., доцент
г. Бирск, ФГБОУ ВО Бирский филиал БашГУ

Реализация свойства интерактивности - это основное требование, которое предъявляется к современным компьютерным обучающим программам. Под интерактивным режимом обучения понимается диалоговый режим работы субъектов образовательного процесса, предполагающий активное взаимодействие обучающегося с системой, имитирующей деятельность педагога через различные средства обучения, контроля, навигации. Для разработки интерактивных обучающих программ в настоящее время используются различные программные среды, но наиболее широкие возможности для реализации именно свойства интерактивности предоставляют современные визуальные среды программирования. В данной статье описывается технология разработки интерактивных элементов обучающих программ в среде Visual Studio на языке программирования Си # [1,2] на примере создания приложения по изучению способов кодирования графической информации. При работе с приложением учащиеся получают возможность в процессе интерактивного диалога понять основные принципы кодирования графической информации на основе визуальных моделей.

Приложение состоит из титульной формы (рис.1) и четырёх разделов-глав, каждый из которых, кроме четвёртого, включает три составляющие: *теория*, *тренажёр*, *контроль*. В четвёртом разделе в силу его специфики отсутствует контроль. Далее разделы будем называть ходом программы (первым, вторым и т.д.), а составляющие – стадиями (теория, тренажер, контроль).

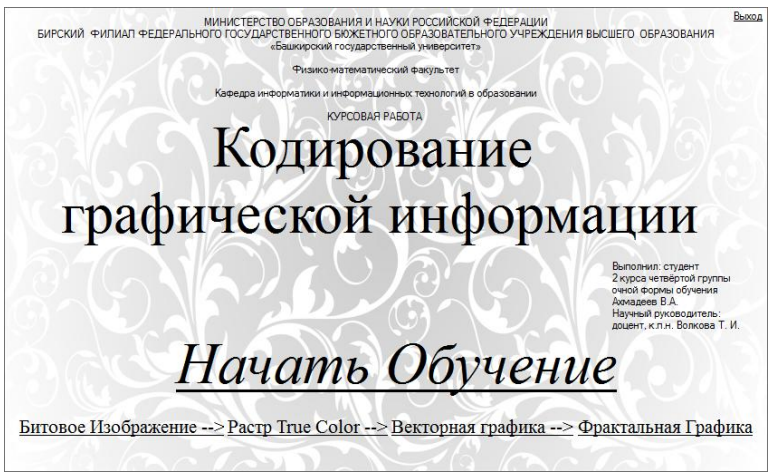

Рис. 1. Титульная форма приложения

Для обеспечения интерактивности и, как следствие, более успешного усвоения теоретического материала, теория и тренажер чередуются. Контроль реализован в форме тестирования, к нему обучаемый переходит после успешного завершения стадий теории и тренажера. При неудовлетворительном прохождении тестирования теоретический материал считается неусвоенным, и обучающийся в принудительном порядке отправляется со стадии тестирования на стадию усвоения теоретического материала.

Во время выполнения любой стадии каждого хода, кроме четвёртого, для удобства обучающегося возможен переход на следующий ход. Из всех стадий любого активного хода возможен переход на титульный лист программы и повторение стадий. Возможен и переход из титульной формы к любой конкретной главе (форме) для повторения материала только данной главы.

Структурная схема разработанного приложения изображена на рисунке 2. Отдельные модули программы, каждый из которых имеет свою *форму*, представлены в виде скруглённых прямоугольников. Стрелками с «уступом» отображены переходы между модулями в соответствии со сценарием. «Прямыми» стрелками – возможные переходы помимо выполнения основного сценария.

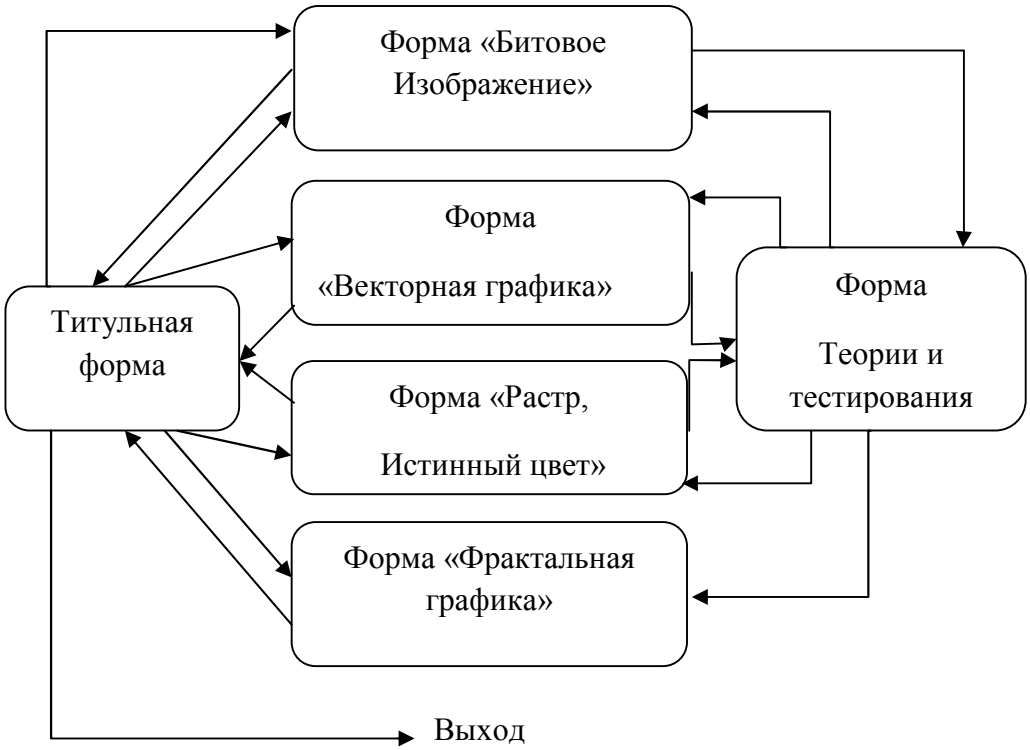

Рис. 2. Структурная схема приложения

Основными средствами обеспечения интерактивности обучения служат обработчики событий «щелчков таймера» (timer_tick) и форма вывода теории и вопросов для контроля.

Все интерактивные узлы программы можно условно разделить на три вида:

1. демонстрационные;
2. служебные (тренировочные);
3. контролирующие.

Демонстрационные узлы служат для указания обучаемому способов использования модулей приложения. Один из них – движение указателя «мыши» по элементам управления с попутной генерацией событий нажатия кнопок «мыши» или имитацией ввода с клавиатуры.

Алгоритм узлов демонстрации состоит из набора переменных для задания координат мыши относительно формы и обработчика события *timer_tick,* обеспечивающего своевременное изменение координат, генерации событий нажатия кнопок, «набора» текста и чисел и других событий *timer_tick*, которые дополнительно указывают обучаемому на происходящее.

Алгоритм генерации события *dataGridView2_CellClick* имеет следующее строение:

```
DataGridViewCellEventArgs ez = new DataGridViewCellEventArgs(c, r);
dataGridView2[ez.ColumnIndex, ez.RowIndex].Style.BackColor =Color.Red;
dataGridView2_CellClick(sender, ez);
```

Где *c* и *r* – номера столбца и строки соответственно, *sender* – стандартная переменная типа данных *object*.

В основе данных алгоритмов лежат элементы *ветвления*, условиями которых являются переменные – счетчики направления и текущего индекса массива строковых значений, содержащего теоретический материал. В теле ветвлений выполняются операторы изменения координат, счетчиков направления и управления вспомогательными обработчиками. Указатель мыши перемещается путём задания новой пары координат в конце обработчика данного события у свойства «Позиция» класса «Курсор» текущей позиции относительно элемента управления DataGridView:

```
Cursor.Position = dataGridView2.PointToScreen (new Point(Convert.ToInt32(x), Convert.ToInt32(y)));
```

Всего на одной форме используется до четырех обработчиков *timer_tick*.

Тренировочные алгоритмы активно используют такие элементы управления, как *DataGridView, TextBox, NumericUpDown, PictureBox, Button*. Главной целью данных алгоритмов является объяснение материала на наглядных примерах и визуализация действий пользователя при выполнении поставленной задачи.

Основой интерактивности здесь являются событийные процедуры, инициируемые действиями обучаемого. Вот некоторые из них (где *N* – номер элемента):

- *textBoxN_KeyPress* – реагирует на нажатие клавиш обучаемым при наборе текста в поле ввода. Необходим в случаях моментального (интерактивного) реагирования программы на изменения, вносимые пользователем;
- *dataGridViewN_CellClick* – реагирует на «клики» мышкой по ячейкам данного элемента управления. Используется алгоритмом для выбора ячейки, которой нужно задать цвет, и, как следствие, создания на базе данного элемента пиксельной либо числовой матрицы цветов, необходимой для понимания темы и выполнения заданий;
- *linkLabelN_LinkClicked* – реагирует на «клики» мышкой по элементу *linkLabel1.* В тренировочных формах служит для окончания работы с текущей главой и перехода либо к следующей, либо на титульную форму;
- *buttonN_Click* – реагирует на нажатия кнопок, для инициирования других событий или выполнения различных алгоритмов.
- *numericUpDownN_ValueChanged* – реагирует на изменения значений элемента. Служит для мгновенного отображения изменений цвета.

Также повсеместно используется событие загрузки *FormN_Load*. В нем явно прописываются все необходимые действия, которые должны произойти во время загрузки формы, такие, как установка необходимых свойств форме и элементам управления, объявление и инициализация переменных, объявление и заполнение массивов, инициирование других событий и запуск служебной формы теоретического обучения и контроля.

Алгоритм синхронизации изменений двух элементов *DataGridView* формы «Битовое изображение» находится в обработчике событий *dataGridView2_CellClick и* выглядит следующим образом:

```
If (dataGridView2[e.ColumnIndex, .RowIndex].Value.ToString()== "0")
{  dataGridView2[e.ColumnIndex, e.RowIndex].Value = "1";
  dataGridView2[e.ColumnIndex, e.RowIndex].Style.ForeColor = Color.Red;
  dataGridView1[e.ColumnIndex, e.RowIndex].Style.BackColor
= Color.Black;}
else {  dataGridView2[e.ColumnIndex, e.RowIndex].Value = "0";
dataGridView2[e.ColumnIndex, e.RowIndex].Style.ForeColor
=Color.Black;
dataGridView1[e.ColumnIndex, e.RowIndex].Style.BackColor
= Color.White;}
```

Специально созданный класс *ColorConvert* служит для конвертации значения цвета из цветовой модели RGB в цветовые модели CMYK и HSB. Используется при работе с формой «Растр Истинный цвет». Он включает в себя одноимённые *структуры* для хранения цвета моделей RGB, CMYK, HSB. Метод *RGBtoHSB* данного класса обеспечивает конвертацию из цветовой модели RGB в HSB. Метод *RGBtoCMYK* обеспечивает конвертацию из модели RGB в CMYK. Методы данного класса используют общепринятые математические алгоритмы конвертации значений цветов. Входными параметрами и возвращаемыми значениями являются соответствующие структуры.

Для примера приведём метод конвертации *RGBtoCMYK:*

```
public static CMYK RGBtoCMYK(RGB Color)
{
CMYK C_M_Y_K = new CMYK();
double R, G, B;
R = (double)Color.Red;
G = (double)Color.Green;
B = (double)Color.Blue;
R = 1.0 - (R / 255.0);
G = 1.0 - (G / 255.0);
B = 1.0 - (B / 255.0);
double C, M, Y, K;
if (R < G)
K = R;
else
K = G;
if (B < K)
K = B;
C = (R - K) / (1.0 - K);
M = (G - K) / (1.0 - K);
Y = (B - K) / (1.0 - K);
if (C.ToString() == "NaN")
C = 0;
if (M.ToString() == "NaN")
M = 0;
if (Y.ToString() == "NaN")
Y = 0;
C_M_Y_K.C = (C * 100) /*+ 0.5*/;
C_M_Y_K.M = (M * 100) /*+ 0.5*/;
C_M_Y_K.Y = (Y * 100) /*+ 0.5*/;
C_M_Y_K.K = (K * 100) /*+ 0.5*/;
return C_M_Y_K; }
```

Методы *Julia, MandelbrotSet, Draw* служат для построения изображения в элементе управления PictureBox формы «Фрактальная графика» фрактала Жюлиа, Множества Мандельброта и фрактала «Крест Ньютона» соответственно.

Данные методы, используя широко распространённые циклические итеративные алгоритмы, формируют точечный рисунок, относительно размеров *PictureBox*-а, являющийся объектом класса *BitMap*. После чего этот объект передаётся в качестве изображения в элемент *PictureBox*.

Выбор метода происходит посредством изменения значения в элементе управления *ComboBox*. После выбора названия в элементе управления *pictureBox2* отображается итеративная формула, а в другие элементы управления передаются начальные параметры, уже определённые для «правильного» построения точечного рисунка.

Контролирующие узлы, с одной стороны, всегда присутствуют на форме и следят за тем, что и как изменяет обучаемый. И, при правильной последовательности действий, инициируют завершение текущей и передачу новой задачи для решения. С другой стороны, крупный узел контроля находится на форме тестирования и следит за тем, как обучаемый отвечает на вопросы теста. При неправильном ответе теоретические сведения считаются неусвоенными, вследствие чего стадия тестирования автоматически завершается и пользователь принудительно направляется на начальную стадию прохождения обучения. Данные узлы использую те же обработчики событий, что и вышеперечисленные.

В качестве примера реализации интерфейса рассмотрим несколько форм. На форме «Векторная графика» (рис. 3) можно поэкспериментировать с такими элементами векторной графики как линия, дуга, эллипс и закрашенный эллипс.

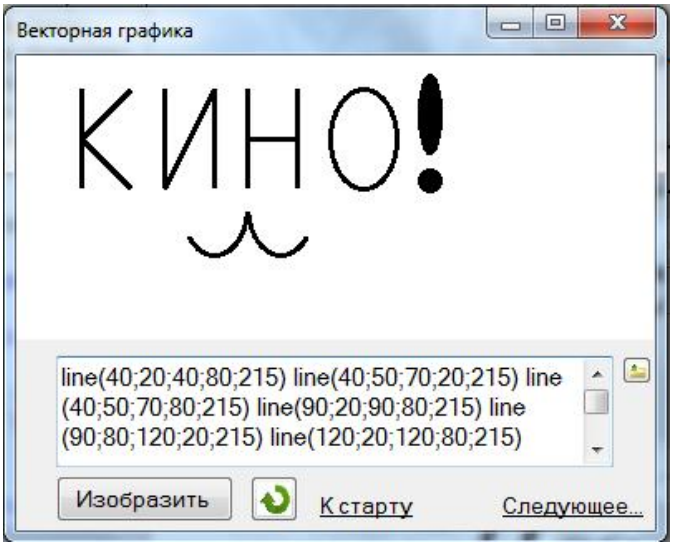

Рис. 3. Изучение векторной графики

У каждого элемента возможно в численном виде задавать почти любые координаты и цвет, ограниченный 216 цветной палитрой. Форма содержит два элемента управления LinkLabel, два TextBox-а для задания элементов векторной графики и вывода подсказки синтаксиса, один PictureBox собственно для отображения элементов векторной графики и элемент Button для выполнения набранных векторных элементов. Имеется также справка по синтаксису задания фигур.

Алгоритм исполнения считывает записи из поля ввода и воспроизводит данные графические примитивы в элементе *PictureBox*. Прорисовка примитивов выполняется стандартными возможностями библиотеки с подключением пространства имён *System.Drawing.Drawing2D*. Все элементы управления рассчитаны на изменение размеров формы и динамически закреплены по координатам.

На форме «Фрактальная графика» реализовано изучение технологии кодирования фрактального изображения (рис.4).

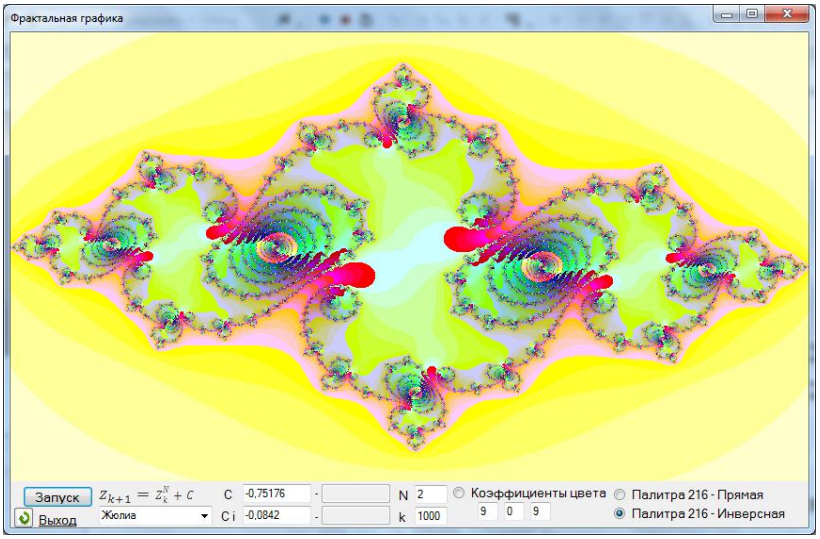

Рис. 4. Визуализация фрактального изображения

Вместо перехода на другую форму по элементу LinkLabel с текстом «Следующее…», размещён тот же элемент, но для выхода и с соответствующим текстом. Кроме того, пользователю предоставляется возможность выбора оттенка, в который будет окрашено фрактальное изображение, путём выбора типа палитры или ручного задания коэффициентов при помощи элементов RadioButton и TextBox.

Также на форме присутствуют такие элементы как Button, для запуска процесса построения фрактала; ComboBox для выбора формулы фрактала; PictureBox, для отображения формулы фрактала; несколько TextBox- ов, для задания коэффициентов формул; несколько меток Label и всё выше перечисленное закреплено на элементе Panel.

Изображение строится на элементе PictureBox, на котором находится элемент ProgressBar, служащий для визуализации процесса построения фрактального изображения.

Все элементы рассчитаны на изменение размеров формы и динамически закреплены по координатам. После прохождения теории обучаемому даётся возможность длительной самостоятельной работы с данной формой. Ограничением является лишь весьма скудный запас формул для построения фракталов.

На служебной форме реализовано отображение теории для каждой из форм, а также проведение тестового контроля для каждой из форм, кроме формы «Фрактальная графика». Присутствуют такие элементы управления, как Button, четыре элемента RadioButton, расположенных поверх элемента TextBox, в который выводится теоретический материал или текст заданий.

Как уже было сказано, теоретический материал, и вопросы теста с вариантами ответов хранятся в массивах. Причём теория хранится в массивах, инициализируемых при загрузке рабочей формы, т.е. именно той главы, на которой специализирована форма. А тестовые вопросы хранятся в массивах, принадлежащих собственно тестовой форме.

При проведении тестирования работает алгоритм из взаимоисключающих вложенных *ветвлений*, в условии которых, выясняется, перешёл ли процесс обучения в стадию тестирования, и в соответствии с какой именно главой должны загрузиться вопросы для проведения тестирования.

Литература

1. Климов Л. П. К49 C#. Советы программистам. — СПб.: БХВ-Петербург, 2008. — 544 с: ил. + CD-ROM
2. Рубанцев Валерий, Программирование на языке C# 5.0: Компьютерная графика. — RVGames 2014. - 94 с.