Лабораторная работа №6

Класс UserControl. Создание пользовательских элементов управления

1) Класс UserControl

Пакет .NET Framework, начиная с версии 3.0, содержит класс UserControl, который является удобным для создания пользовательских элементов управления. Для создания элемента необходимо выбрать пункт меню Проект->Добавить класс, затем в появившемся окне (Рисунок 1) выбрать из списка универсальных шаблонов пункт «Элементы Visual C#», а в списке опций «Пользовательский элемент управления». Затем необходимо в поле «Имя» ввести имя класса элемента управления и нажать кнопку добавить.

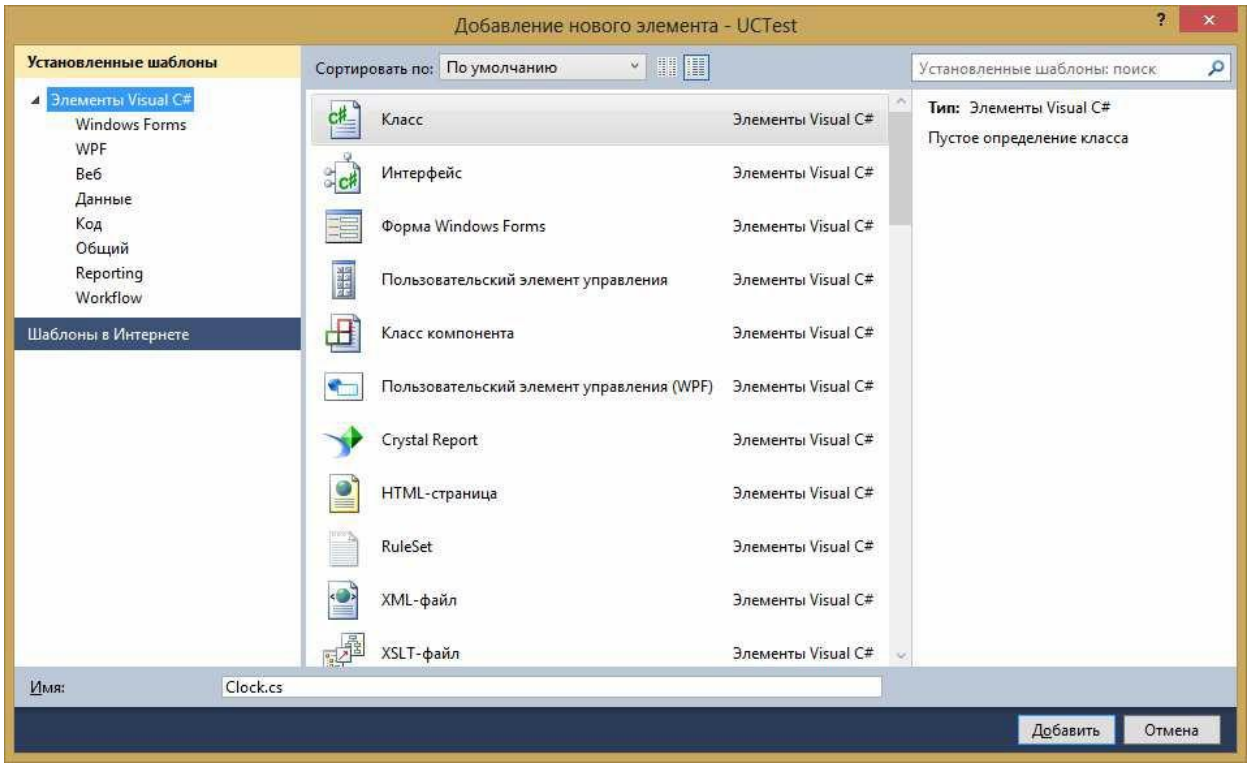

Рисунок 1 – Создание пользовательского элемента управления

Среда разработки автоматически сгенерирует пустой элемент, который можно использовать как форму и добавить в него другие элементы. Таким образом осуществляется создание составных элементов управления. Подробнее о составных элементах можно узнать из ресурса MSDN

http://msdn.microsoft.com/ru-ru/library/a6h7e207.aspx.

Добавление пользовательского элемента на форму производится также как и добавление стандартного компонента – через Панель элементов. Более интересным является случай оригинального элемента управления, при построении которого не используются стандартные элементы.

2) Класс Graphics

Для прорисовки пользовательского компонента необходимо переопределить метод protected override void OnPaint(PaintEventArgs e). Из аргумента e можно получить экземпляр класса Graphics, который используется для рисования на компоненте (http://msdn.microsoft.com/ru-ru/library/system.drawing.graphics.aspx). Рассмотрим пример:

```
protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    Graphics g = e.Graphics;
    g.DrawLine(Pens.Black, 0, 0, Width, Height);
}
```

В этом примере сначала вызывается метод OnPaint базового класса, затем производится получение ссылки на Экземпляр класса Graphics элемента, и рисуется линия черным пером (Pens.Black) от верхнего левого угла до нижнего правого.

Для определения размеров компонента обычно переопределяют метод protected override void OnResize(EventArgs e).

3) Реализация пользовательского компонента Часы

Рассмотрим пример создания компонента Clock (Часы). Для этого воспользуйтесь пунктом меню «Проект->Добавить класс» и добавьте пользовательский элемент правления с именем «Clock.cs». В класс добавьте следующий код:

```
public partial class Clock : UserControl
{
    private Timer mTimer; //Таймер, который генерирует событие перерисовки
    private Pen mSimplePen; //Хранит экземпляр Пера
    private Color mPointerColor; //Цвет указателей стрелок
    private int mRadius; //Радиус часов
```

```
    private Point mCenter; //Координты центра компонента
    public Clock() //Конструктор по умолчанию
    {
        InitializeComponent();

        //Создание таймера
        mTimer = new Timer();
        mTimer.Interval = 1000;
        mTimer.Tick += new EventHandler(TimerTick);
        mTimer.Start();

        //Создание пера
        mSimplePen = new Pen(Brushes.DarkGray, 1.5f);
        mPointerColor = Color.Black;
    }

    private void TimerTick(object sender, EventArgs e)
    {
        Invalidate();
    }

    protected override void OnPaint(PaintEventArgs e)
    {
        base.OnPaint(e);
        Graphics g = e.Graphics;
        g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;

        //Рисование круга
        g.DrawEllipse(mSimplePen, new Rectangle(mCenter.X - mRadius, mCenter.Y -
mRadius, mRadius * 2, mRadius * 2));
        g.FillEllipse(Brushes.Black, new Rectangle(mCenter.X - 3, mCenter.Y - 3,
6, 6));

        //Рисование рисок
        for (double angle = 0; angle < 2 * Math.PI; angle += Math.PI / 6.0f)
        {
            double a = angle - Math.PI / 2;
            g.DrawLine(mSimplePen, (float)(mRadius * Math.Cos(a)) + mCenter.X,
                (float)(mRadius * Math.Sin(a)) + mCenter.Y,
                (float)((mRadius - 5) * Math.Cos(a)) + mCenter.X,
                (float)((mRadius - 5) * Math.Sin(a)) + mCenter.Y);
        }

        //Получение текущего времени
        DateTime currentTime = DateTime.Now;

        //Секундная стрелка
        float sa = (float)(Math.PI * currentTime.Second / 30 - Math.PI / 2.0);
        g.DrawLine(new Pen(mPointerColor, 1f),
            (float)(-(20) * Math.Cos(sa)) + mCenter.X,
            (float)(-(20) * Math.Sin(sa)) + mCenter.Y,
            (float)((mRadius - 10) * Math.Cos(sa)) + mCenter.X,
            (float)((mRadius - 10) * Math.Sin(sa)) + mCenter.Y);
        //Минутная стрелка
        float ma = (float)(Math.PI * currentTime.Minute / 30 - Math.PI / 2.0);
        g.DrawLine(new Pen(mPointerColor, 2f), mCenter.X, mCenter.Y,
            (float)((mRadius - 10) * Math.Cos(ma)) + mCenter.X,
            (float)((mRadius - 10) * Math.Sin(ma)) + mCenter.Y);
        //Часовая стрелка
        float ha = (float)(Math.PI * currentTime.Hour / 6 - Math.PI / 2.0) +
(float)(Math.PI * currentTime.Minute / 360);
        g.DrawLine(new Pen(mPointerColor, 3f), mCenter.X, mCenter.Y,
            (float)((mRadius - 15) * Math.Cos(ha)) + mCenter.X,
            (float)((mRadius - 15) * Math.Sin(ha)) + mCenter.Y);
    }

    protected override void OnResize(EventArgs e)
    {
```

```
        base.OnResize(e);

        //Определение координат центра часов
        mCenter.X = ClientRectangle.Width / 2;
        mCenter.Y = ClientRectangle.Height / 2;

        //Определение радиуа циферблата
        if (mCenter.X > mCenter.Y)
            mRadius = mCenter.Y;
        else
            mRadius = mCenter.X;
        mRadius -= 3;
    }
}
```

В методе onResize производится определение координат центра часов и определяется радиус циферблата, который равен наименьшему из значений полувысоты и полуширины компонента. При перерисовке компонента (метод onPaint) сначала рисуется циферблат часов, затем производится получение текущего значения времени, и рисуются секундная, минутная и часовая стрелки. Вид компонента на форме во время выполнения программы показан на рисунке 2.

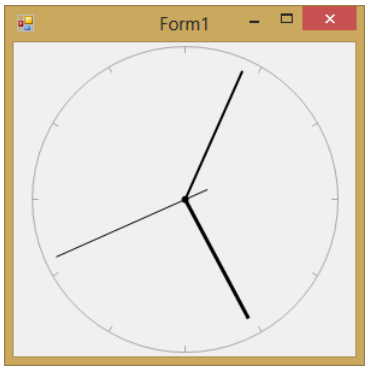

Рисунок 2 – Компонент «Часы»

4) Задания к лабораторной работе №6

1) Реализовать класс LineChart, позволяющий отображать зависимости вида f(x). Для хранения данных использовать стандартные классы PointF, и List<PointF>. Предусмотреть возможность изменения цвета зависимости, автоматического масштабирования графика, изменение подписей осей абсцисс и ординат.

2) Реализовать класс PieChart, позволяющий отображать круговую диаграмму. Для хранения данных создать пользовательский класс, который

состоит из полей Цвет (Color) и значение (int). Для хранения списка использовать класс List<>. Предусмотреть возможность и изменения подписи оси ординат.

3) Реализовать класс BarChart, позволяющий отображать гистограмму. Для хранения данных создать пользовательский класс, который состоит из полей Цвет (Color) и значение (int). Для хранения списка использовать класс List<>. Предусмотреть возможность автоматического масштабирования и изменения подписи оси ординат.

4) Написать графический компонент, который отображает текущее время в виде семисегментного индикатора.

5) Написать класс Potentiometer, представляющий собой круглую плашку, которая может вращаться вокруг своей оси на угол порядка 300 градусов. Рядом с плашкой должны располагаться отметки. Пользователь может вращать плашку, а у компонента должно меняться свойство value в зависимости от угла поворота плашки.

6) Реализовать элемент управления, имеющий внешний вид градусника. Градусник должен иметь отградуированную шкалу, со значениями от Minimum до Maximum (реализованные в классе свойства). Свойство Value задаёт высоту цветного столбика внутри термометра, отображающее текущее значение. Свойство Text для элемента управления должно задавать текст надписи (заголовка) над градусником. Усложненный вариант задания: разработанный элемент управления должен быть использован в приложении, для отображения с его помощью текущий уровень загрузки центрального процессора.

7) Реализовать класс NuPogodi, позволяющий отображать экран портативной игры "Ну, погоди!" ("Электроника ИМ-02"). Детальное описание задания в NuPogodi.zip.

8) Реализовать класс (пользовательский элемент управления) TicTacToe, позволяющий отображать поле для игры в «Крестики-нолики». При изменении размеров элемента управления, поле так же должно масштабироваться. В классе должно быть реализовано свойство Values, являющееся числовым массивом [3, 3], значения в котором задают содержимое ячеек игрового поля (0 – ячейка пустая, 1 – «крестик», 2 – «нолик»). Должно быть реализовано

событие клика по ячейке игрового поля, при этом обработчику события должны передаваться дополнительные параметры – координаты ячейки по которой кликнул пользователь (например, x = 1, y = 1 – если пользователь кликнул по центральной ячейке).

9) Создать элемент управления, похожий на элемент Label, но выводящий указанный в свойстве Text текст в виде шрифта Брайля. Размер шрифта должен регулироваться, при изменении размеров элемента управления непомещающиеся в строке буквы переносятся на новую строку.

5) Контрольные вопросы

1) Для чего предназначен класс UserControl?

2) Какие основные методы рисования графических примитивов содержит класс Graphics?

3) Опишите класс Pen. Для чего он применяется?

4) Для чего применяется класс Brush? Опишите классы, наследующие абстрактный класс Brush.

5) Как создать произвольный цвет в виде структуры Color?

6) Как вызвать принудительную перерисовку компонента?

7) Для чего необходима возможность создания пользовательских элементов управления? Приведите примеры таких ситуаций.