



Micro  
Framework


# WPF в .NET Micro Framework

Алексеев Пётр  
Доцент кафедры Радиоэлектронных систем управления (И4) БГТУ «Военмех»

MCSD.NET

[www.netmf.ru](http://www.netmf.ru)

# План

- ▶ Использование графических примитивов
  - ▶ Реализация WPF в .NET Micro Framework
  - ▶ Элементы управления
  - ▶ Использование сенсорных экранов
- 

# Использование графических примитивов

»» Часть 1

# Простейшая графика в .NET MF

- ▶ Основа – работа с изображениями (класс `Microsoft.SPOT.Graphics.Bitmap`)
- ▶ Основные возможности
  - Графические примитивы
  - Текст
  - Цвета, кисти
  - Прозрачность
- ▶ Типовая работа с изображением
  - Создание экземпляра `Bitmap`, задание размеров
  - Рисование элементов изображения за счёт использования методов объекта `Bitmap`
  - Вывод полученного изображения на экран

# Класс Bitmap

- ▶ Конструктор
  - На основании размеров
  - На основании массива байтов и типа изображения (Bmp, Gif, Jpeg, TinyCLRBitmap)
- ▶ Свойства
  - Height
  - Width
- ▶ Методы
  - Графические примитивы
    - Line, DrawLine, Circle, DrawEllipse, DrawRectangle
  - Текст
    - DrawText, DrawTextInRect
  - Изображения
    - DrawImage
  - Общие
    - Clear, Flush

# Пример работы с изображением

```
Bitmap bmp = new Bitmap(Bitmap.MaxWidth,  
    Bitmap.MaxHeight);  
bmp.DrawLine(  
    Color.White, // Цвет линии  
    5, // Толщина линии  
    0, 0, // Координаты начальной точки  
    bmp.Width, bmp.Height); // Ширина и высота линии  
// Вывод изображения на экран  
bmp.Flush(  
    // Левая координата  
    (bmp.Width - Bitmap.MaxWidth) / 2,  
    // Верхняя координата  
    (bmp.Height - Bitmap.MaxHeight) / 2,  
    bmp.Width, // Ширина  
    bmp.Height); // Высота
```

Толщина линии всегда 1, даже если просим 5! ☹



# Цвета

- ▶ Перечислимый тип `Color` – только два цвета: `Color.Black` и `Color.White`
- ▶ Класс `Colors` – уже шесть цветов:
  - `Black #000000`, `Blue #0000FF`, `Gray #808080`, `Green #008000`, `Red #FF0000`, `White #FFFFFF`
- ▶ Остальные цвета следует получать так:
  - `ColorUtility.ColorFromRGB(0xFF, 0xFF, 0x00)`
- ▶ Можно получить составляющие из готового цвета:
  - `ColorUtility.GetRValue`
  - `ColorUtility.GetGValue`
  - `ColorUtility.GetBValue`

# Прямоугольники с прозрачностью

```
Color[] colors = new Color[] {
    ColorUtility.ColorFromRGB(0xFF, 0, 0), // red
    ColorUtility.ColorFromRGB(0, 0xFF, 0), // green
    ColorUtility.ColorFromRGB(0, 0, 0xFF) // blue
};
for (int i = 0; i < colors.Length; i++)
{
    Color color = colors[i];
    bmp.DrawRectangle(color, // Цвет фона
        0, // Толщина линий прямоугольника
        50 + i * 20, 50 + i * 20, // x и y левого
    верхнего угла
        200, 100, // Ширина и высота
        0, 0, // x и y радиуса углового скругления
        color, // Цвет начала градиента
        0, 0, // Координаты начала градиента
        color, // Цвет окончания градиента
        0, 0, // Координаты окончания градиента
        64); // Прямоугольник полупрозрачный
}
```



# Прямоугольник с градиентом

```
bmp.DrawRectangle(Color.White, // Цвет фона
    1, // Толщина линий прямоугольника
    100, 100, // x и y левого верхнего угла
    200, 100, // Ширина и высота
    0, 0, // x и y радиуса углового скругления
    Colors.White, // Цвет начала градиента
    100, 100, // Координаты начала градиента
    Colors.Red, // Цвет окончания градиента
    100 + 200, 100 + 100, // Координаты окончания
градиента
    Bitmap.Opaque); // Прямоугольник
непрозрачный
```



# Изображение

```
Bitmap globe =  
    Resources.GetBitmap(Resources.BitmapResources.Globe);  
bmp.DrawImage(  
    100, 50, // Координаты изображения  
    globe, // Изображение  
    0, 0, // Координаты отображаемой  
    // части исходного изображения  
    globe.Width, // Ширина отображаемой  
    // части исходного изображения  
    globe.Height, // Высота отображаемой  
    // части исходного изображения  
    Bitmap.Opaque); // Изображение непрозрачно
```



# Текст

- ▶ Используются шрифты TinyFnt
- ▶ Оформление шрифта определяется заранее и не доступно для изменения
- ▶ Можно использовать обычные шрифты TrueType, преобразованные с помощью утилиты TFConvert
- ▶ Для отображения текста на русском языке необходимо освоить эту утилиту

```
Font font =  
  
Resources.GetFont(Resources.FontResources.Consolas23);  
bmp.DrawText("Hello from .NET MF.", // Текст  
    font, // Шрифт  
    Color.White, // Цвет  
    20, 20); // x и y верхнего левого угла
```



# Поддержка шрифтов

- ▶ Особенности обычного шрифта OpenType
  - Является векторным
  - Может содержать избыточный набор символов
  - Может занимать 300–400 кб
- ▶ Шрифт TinyFNT (TFConvert.exe)
  - Являются растровыми
  - Содержат только нужные символы
  - Уже содержат форматирование
  - Занимает 10–30 кб

# Возможности DrawTextInRect

- ▶ Отображение текста внутри рамки
- ▶ Перенос по словам `Bitmap.DT_WordWrap`
- ▶ Выравнивание текста внутри прямоугольника
  - По левому краю `Bitmap.DT_AlignmentLeft`
  - По центру `Bitmap.DT_AlignmentCenter`
  - По правому краю `Bitmap.DT_AlignmentRight`
- ▶ Постраничный вывод текста
  - Используется перегруженный вариант, удаляющий из строки показанную часть
  - Достаточно вызывать этот метод до тех пор, пока возвращаемое значение не станет равно `true`

# Демо

- ▶ Работа с изображением
  - Графические примитивы
  - Создание шрифта
  - Текст с постраничным выводом


# Выводы по возможностям простейшей графики

- ▶ Всё завязано на изображения
- ▶ Удобно организовать Z-буфер
- ▶ Целесообразно использовать, если не хватает производительности на WPF
- ▶ Элементы управления отсутствуют
- ▶ Для приложений с графическим интерфейсом пользователя лучше использовать WPF

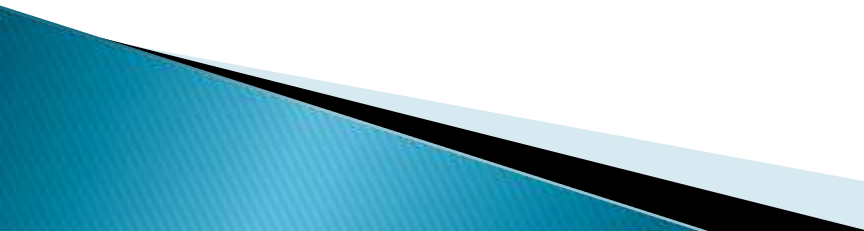
# Реализация WPF в .NET Micro Framework

»» Часть 2

# Основные принципы WPF

- ▶ Декларативное описание в XAML
  - ▶ Вложенность элементов управления
  - ▶ Поддержка векторной графики
  - ▶ Независимость от устройства отображения информации
- 

# Реализация WPF в .NET MF

- ▶ XAML – не поддерживается
  - ▶ Вложенность элементов управления – обеспечена
  - ▶ Поддержка векторной графики – не полностью
  - ▶ Независимость от устройства отображения информации – вывод только на экран
  - ▶ Реализован очень ограниченный набор элементов управления
  - ▶ Нет перекрывающихся окон
- 

# Создание приложения WPF

- ▶ На самом деле, это простейшее оконное приложение
- ▶ Для его создания существует стандартный шаблон

# Основные классы WPF

## ▶ Класс Application

- Объект этого класса – само приложение
- Может содержать одно или несколько окон


## ▶ Класс Window

- Это окно, в которое можно помещать элементы управления
- Может принимать фокус управления

## ▶ Типы элементов управления

- Статические (Text Bitmap)
- Контейнеры (Border, ListBoxItem)
- Панели (Canvas, Panel)

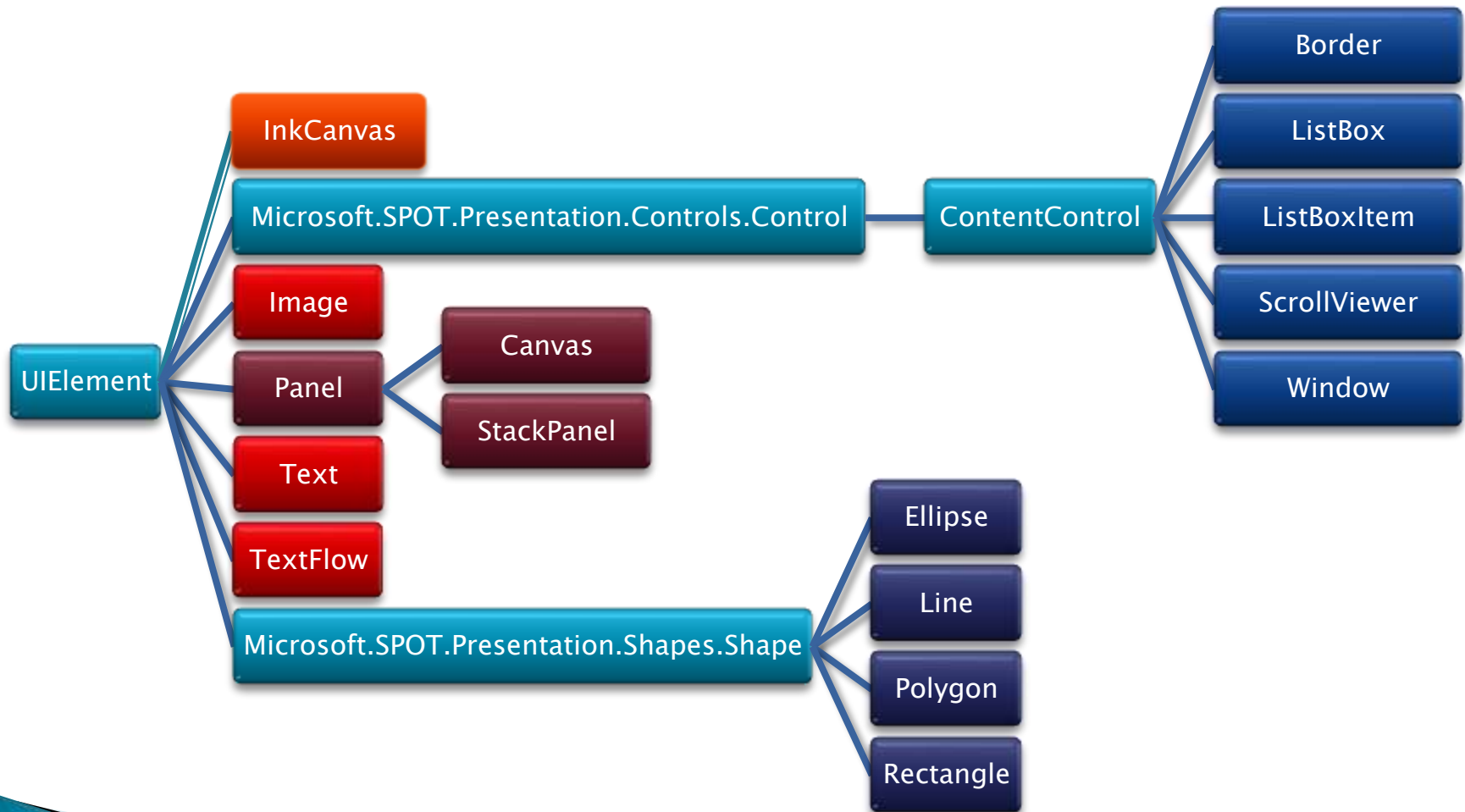
# Как создать приложение WPF?

- ▶ Класс приложения WPF должен быть наследником класса `Microsoft.SPOT.Application`
  - ▶ Создать экземпляр класса приложения
  - ▶ Создать экземпляр главного окна приложения
  - ▶ Запустить экземпляр приложения с помощью метода `Run` с параметром главного окна
- 

# Элементы управления

»» Часть 3

# Иерархия классов элементов управления



# Использование панелей

## ▶ Panel

- Допускается размещение нескольких дочерних элементов управления
- Нет возможности указать координаты расположения элементов управления, можно только задать выравнивание
- Элементы управления перекрывают друг друга
- Размеры панели определяются размерами дочерних элементов управления

## ▶ StackPanel

- Делает всё то же самое, что и Panel (является наследником этого класса)
- Дочерние элементы управления выстраиваются по горизонтали или вертикали (определяется свойством Orientation)
- Возможность выравнивания каждого из дочерних элементов управления в отдельности по горизонтали и вертикали
- Можно установить поля (Margins) дочерних элементов управления

## ▶ Canvas

- Является наследником класса Panel и может выполнять аналогичные действия
- Есть возможность явно указать координаты расположения дочерних элементов управления

# Демо

- ▶ Выравнивание в Panel, StackPanel и Canvas

# Кисти

- ▶ Позволяют сделать заливку фона элементов управления, сделать заливку содержимого элементов управления
- ▶ Сплошные кисти `SolidColorBrush`
- ▶ Градиентные кисти `LinearGradientBrush`
- ▶ Заливка изображениями `ImageBrush`
  - Можно растянуть изображение без сохранения пропорций
  - Размножить нельзя

# Рамки

- ▶ Рамка – элемент управления наравне с остальными
- ▶ Может добавляться на панели, как изображение или текст
- ▶ Внутри рамки следует добавить элемент управления, который она обрамляет с помощью свойства Child
- ▶ Можно установить толщину границ и указать кисть для заливки
- ▶ Толщина границ может быть разной в разных направлениях

# Демо

- ▶ Выделение текста рамкой с градиентной заливкой

# Отображение текста

- ▶ Простейший вариант – Text
- ▶ TextFlow
  - Отображение текста разным оформлением в пределах одного поля
  - Постраничный вывод текста с обработкой нажатий кнопок
  - Скроллинг текста с обработкой нажатий кнопок
  - Обработка нажатий кнопок осуществляется с помощью указания фокуса кнопок `Buttons.Focus(textFlow)`

# Демо

- ▶ Скроллинг текста

# Скроллинг произвольного элемента управления

## ▶ ScrollView

- Постраничный и построчный скроллинг
- Вертикальный и горизонтальный скроллинг
- Возможность задать шаг скроллинга в пикселях `LineHeight`, `LineWeight`
- Для скроллинга множеств элементов управления следует использовать `Panel`, `StackPanel` или `Canvas`

# Демо

- ▶ Скроллинг изображения

# Создание меню

## ▶ ListBox

- Возможность добавить несколько пунктов типа `ListBoxItem`
- Автоматическая обработка нажатий кнопок для определения выбранного пункта на основе событий
  - `SelectionChanged` – выбран новый пункт с помощью кнопок вверх и вниз
  - `listBox.AddHandler(Buttons.ButtonDownEvent, new ButtonEventHandler(listBox_ButtonDown), false)` – возможность обработки нажатия кнопки `Select`
- Отображение выбранного пункта следует обеспечивать самостоятельно за счёт наследования от класса `ListBoxItem` и переопределения метода `OnIsSelectedChanged`

# Демо

- ▶ Работа с меню

# Разработка собственных элементов управления

- ▶ Выбор наиболее подходящего базового класса
- ▶ Определение размера `MeasureOverride`
- ▶ Собственная отрисовка `OnRender`
- ▶ Перерисовка после изменения свойств `Invalidate`
- ▶ Обработка событий `OnButtonUp`, `OnButtonDown`
  - Разрешение или запрет дальнейшей обработки события `e.Handled`

# Использование сенсорных экранов

»» Часть 4

# Что есть для работы с сенсорными экранами

- ▶ События класса `UIElement`
  - `StylusDown`
  - `StylusUp`
  - `StylusMove`
- ▶ Элемент управления `InkCanvas` – рисование стилусом и распознавание жестов
- ▶ Класс `TouchCollectorConfiguration` – настройка
- ▶ Класс `Touch` – калибровка

# Настройка подсистемы сенсорного экрана

- ▶ Класс TouchCollectorConfiguration
- ▶ Свойства
  - CollectionMethod: Native или Managed
  - CollectionMode: Ink, Gesture, Ink и Gesture
  - SamplingFrequency
  - StylusMoveFrequency
- ▶ Методы
  - GetLastTouchPoint
  - GetSetTouchInfo
  - SetStylusMoveFrequency

# Калибровка TouchPanel

- ▶ Свойство Touch.ActiveTouchPanel
- ▶ Класс TouchPanel
- ▶ СВОЙСТВО
  - Enabled
- ▶ Методы
  - GetCalibrationPointCount (ref int count)
  - GetCalibrationPoint(int index, ref int x, ref int y)
  - StartCalibration()
  - SetCalibration (  
int cCalibrationPoints,  
short[] screenXBuffer,  
short[] screenYBuffer,  
short[] uncalXBuffer,  
short[] uncalYBuffer)

# Демо

- ▶ Перетаскивание элементов управления
- ▶ Рисование стилусом и распознавание жестов

# Выводы по использованию WPF

- ▶ Многих привычных элементов управления не хватает
- ▶ Очень удобно для организации разметки экрана
- ▶ Очень удобно применять при реализации меню, проводника и т.п.
- ▶ Практически готовый интерфейс пользователя для «читалки»
- ▶ Очень хорошая скорость работы
- ▶ Сенсорный экран заметно улучшает интерфейс пользователя

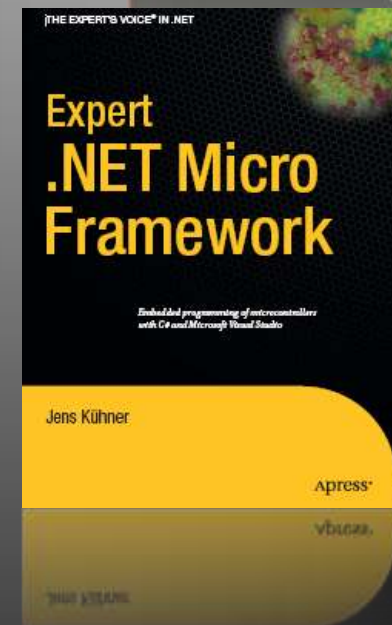
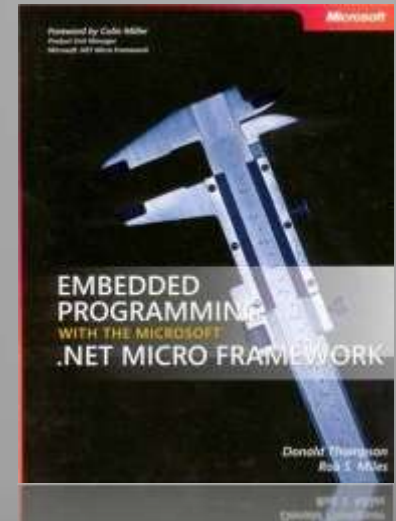
# Дополнительная информация

## ▶ Книги

- Embedded Programming with the Microsoft .NET Micro Framework
- Apress Expert .NET Micro Framework Expert 2008

## ▶ Ссылки

- <http://www.microsoft.com/netmf>
- <http://www.microframework.eu>
- <http://www.netmf.ru>



**Спасибо за внимание!**

