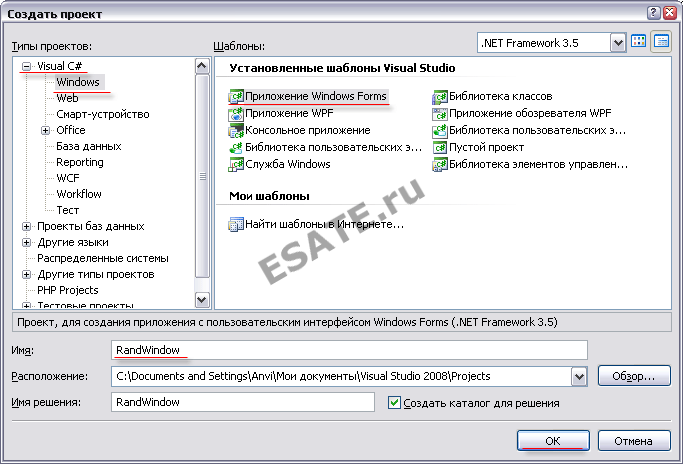
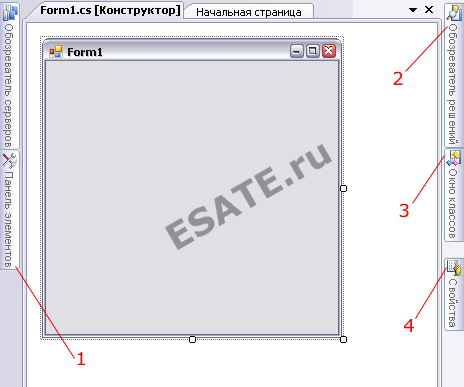
# Создание первого оконного приложения в .NET. "Убегающее окно".

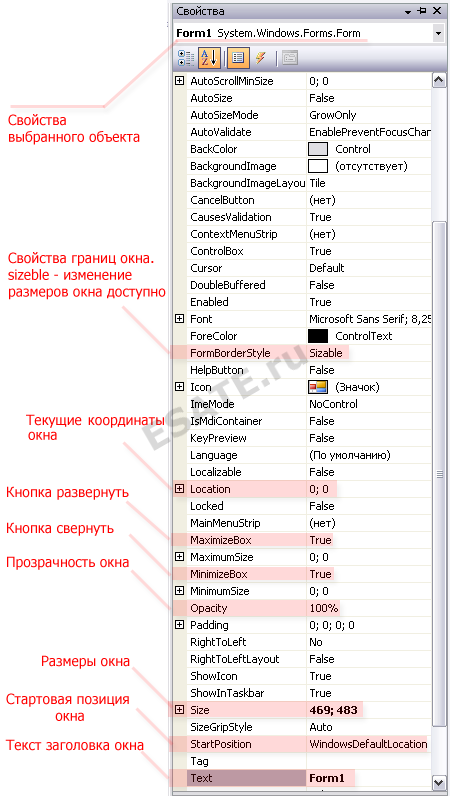
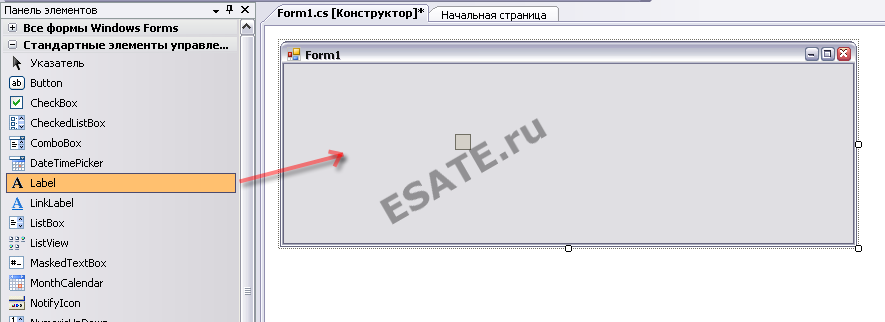
Разрабатываемое приложение будет представлять собой небольшую форму, повторяющую идею давным-давно придуманной программы: на форме будет содержаться вопрос «**Вы стремитесь сделать мир лучше?**». При попытке наведения указателя мыши на кнопку «**Да, конечно!**» окно будет убегать от нас. Нажатие на кнопку «**Нет**» не будет ничем ограничено.

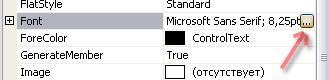
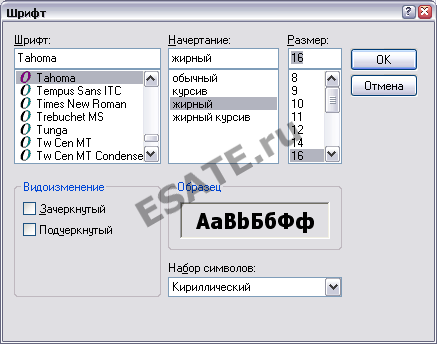
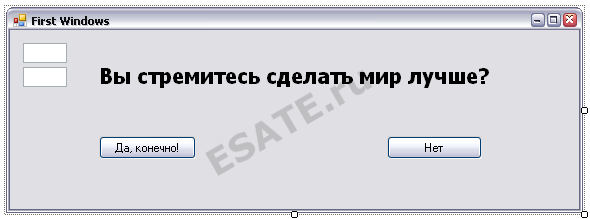
Создавая данную программу, мы рассмотрим основные принципы создания оконных приложений в **C#**, а также методы реализации с помощью них каких-либо конечных целей: в нашем случае - это забавно убегающее окно.   
  
Создайте новый проект, в качестве типа шаблона установите приложение **Windows Forms**, как показано на рисунке 1:  
Рисунок 1. Создание нового проекта.   
Назовите проект **RandWindow** и нажмите кнопку **ОК**.   
  
Рабочее окно **MS Visual Studio** содержит следующие вспомогательные окна (рис. 2).  
Рисунок 2. Вспомогательные окна.   
На рисунке цифрам отмечены:

1. Окно **Toolbox** (Панель элементов управления) - элементы управления вы можете разместить на создаваемой форме.
2. Окно **Solution Explorer** (Обозреватель решений) - здесь вы сможете увидеть следующие узлы: **Properties** - настройки проекта, **Links** (Ссылки) - подключенные к проекту библиотеки, а также созданные и подключенные к проекту файлы исходных кодов (с расширением **.cs**) и подключенные к проекту формы (например, **Form1**).
3. Окно **Class View** (Окно классов) - здесь представлены все созданные в программе классы.
4. Окно **Properties** (Свойства) - выбрав любой элемент управления или даже форму, вы сможете увидеть все параметры данного объекта, а также изменить значения, установленные в них по умолчанию.

Создание оконных приложений сводится к созданию всех необходимых диалоговых окон, а также к размещению на них необходимых элементов. В дальнейшем мы настраиваем обработку событий, создаваемых пользователем, и настраиваем технические аспекты работы программы. В нашем случае сначала разместим все необходимые элементы управления на главной форме, после чего добавим обработчик события перемещения мыши и обработку нажатия кнопок.

#### Добавление новых элементов управления на форму

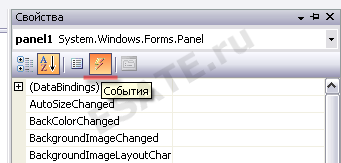
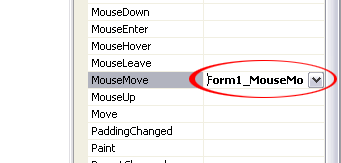
Итак, после того как вы ввели имя проекта, установили необходимый шаблон и нажали кнопку **ОК**, **MS Visual Studio** автоматически создаст каркас оконного приложения, после чего мы сможем добавить на него новые оконные элементы.   
  
Для этого необходимо перетащить необходимый оконный элемент из окна инструментов (**ToolBox**).  
  
Нашему окну потребуется **2** элемента поля для ввода, в которые мы будем выводить координаты указателя мыши, что облегчит нам понимание работы программы.   
  
В центре окна будет находиться надпись, которую мы создадим с помощью элемента **Label**.   
Снизу будут расположены **2** кнопки.   
  
Немного растяните заготовку окна. Если вы нажмете на него правой кнопкой, то откроется контекстное меню. В нем нажмите на пункте свойства, после чего вы сможете изучить различные параметры окна, которые вы можете изменить. На рисунке 3 изображены самые (как правило) необходимые свойства:  
Рисунок 3. Различные свойства окна (формы) в C# .NET.   
Немного растяните заготовку окна и добавьте все необходимые элементы. На рисунке 4 вы можете увидеть их в окне **ToolBox**:  
Рисунок 4. Перетаскивайте необходимые элементы из окна Toolbox на создаваемую форму.

Перейдите в свойства строки **Label1**, где измените текст на «**Вы стремитесь сделать мир лучше?**». Также измените тип шрифта, для этого найдите свойство **Font** (рис. 5).  
Рисунок 5. Свойство Font элемента Label.   
После чего установите тип шрифта **Tahoma**, ширину шрифта **Bold** и размер равный **16** (рис. 6).  
Рисунок 6. Установки шрифта.   
Далее измените текст на кнопках, используя свойство **Text**.   
  
Полученная заготовка окна программы будет выглядеть следующим образом (рис. 7).  
Рисунок 7. Форма будет выглядеть следующим образом.

#### Техническая часть работы программы

1. Сначала мы добавим обработчик события перемещения мыши и реализуем вывод ее координат **x** и **y** в два созданных поля ввода.
2. Далее мы создадим функции обработчики щелчка по каждой из клавиш мыши (особенно усердные пользователи все же смогут попасть по кнопке «**Да, конечно!**»).
3. Далее мы добавим код, реализующий случайное перемещение окна в том случае, если курсор приблизиться к кнопке «**Да, конечно!**».

### Определение перемещения указателя мыши по форме

Щелкните непосредственно на части формы создаваемого приложения (**НЕ** на одном из элементов).   
  
Теперь перейдите к свойствам формы с помощью щелчка правой кнопки мыши **->** контекстное меню **свойства**.   
  
Теперь необходимо перейти к списку возможных событий, которые может получать данное окно. Для этого щелкните на кнопке «**Event**» (события), как показано на рисунке 8:  
Рисунок 8. Переход к списку возможных событий.   
Когда пользователь передвигает указатель мыши по нашему окну, операционная система посылает сообщение программе с текущими координатами указателя. Они-то нам и нужны.   
  
Чтобы назначить обработчик данного события, найдите строку **MouseMove** (рис. 9), после чего сделайте двойной щелчок в поле справа от нее – автоматически добавится обработчик события движения мыши и добавится функция **Form1\_MouseMove** в коде нашей программы.  
Рисунок 9. После двойного щелчка справа от MouseMove появиться данная строка, и автоматически совершится переход к коду функции-обработчика данного события.   
Добавьте в эту функцию 2 строки, чтобы ее код стал выглядеть следующим образом.

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

// ппереводим координату X в строку и записывает в поля ввода

textBox1.Text = e.X.ToString();

// переводим координату Y в строку и записывает в поля ввода

textBox2.Text = e.Y.ToString();

}

Данная функция, обрабатывающая событие перемещения указателя мыши над формой, получает **2** параметра: объект отправитель и экземпляр класса **MouseEventsArgs**, содержащий информацию о координатах указателя мыши и других текущих свойствах.   
  
**textBox1** и **textBox2** это экземпляры класса **textbox**, реализующие управление нашими элементами поля для ввода.   
  
Член данных экземпляров **Text** позволяет установить текст в данных полях.   
  
Таким образом, если теперь откомпилировать программу (**F5**), при перемещении указателя мыши по форме окна мы будем видеть координаты указателя (внутри формы), которые будут непрерывно изменяться.   
  
Теперь вернемся к заготовке нашей формы. Для это щелкните на соответствующей закладке **(Form1.cs [Конструктор])**, как показано на рисунке 10:  
Уроки OpenGL + C#: Переход к конструктору форм C#Рисунок 10.

Переход к конструктору форм C#.   
Сделайте двойной щелчок по первой кнопке: **Visual Studio** автоматически добавит код обработки данной кнопки при нажатии.   
  
Добавьте следующие строки кода:

private void button1\_Click(object sender, EventArgs e)

{

// Вывести сообщение с текстом "Вы усердны"

MessageBox.Show( "Вы усердны!!");

// Завершить приложение

Application.Exit();

}

Теперь снова вернитесь к конструктору и добавьте вторую кнопку, также с помощью двойного щелчка по ней.   
  
Она будет содержать следующий код:

private void button2\_Click(object sender, EventArgs e)

{

// Вывести сообщение, с текстом "Мы не сомневались в вешем безразличии"

// второй параметр - заголовок окна сообщения "Внимание"

// MessageBoxButtons.OK - тип размещаемой кнопки на форме сообщения

// MessageBoxIcon.Information - тип сообщения - будет иметь иконку "информация" и соответствующий звуковой сигнал

MessageBox.Show("Мы не сомневались в вешем безразличии","Внимание", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

Как видите, здесь мы немного усложнили код вызова окна-сообщения, чтобы продемонстрировать то, как оно работает, более подробно. Все параметры передаваемые в функцию **Show** закомментированны в исходном коде.   
  
Теперь нам осталось только реализовать перемещение окна в тот момент, когда мышь приближается к кнопке «**Да, конечно**».   
  
Для этого мы добавим код в функцию:  
  
**private void Form1\_MouseMove(object sender, MouseEventArgs)**  
  
Принцип очень прост: получая координаты движения мыши, мы проверяем не входят ли они в квадрат, очерчивающий нашу кнопку с небольшим запасом. Если да, то мы генерируем 2 случайных числа, которые будут использованы для перемещения окна.   
  
Мы бы могли просто отслеживать сообщение о наведении указателя мыши на кнопку, но оно приходит с заметной задержкой, в связи с чем пользователь без особого труда нажмет на кнопку да, поэтому мы будем просто вычислять попадание курсора в зону вокруг кнопки.   
  
Также нам понадобится объявить несколько "**рабочих**" переменных, которые мы будем в дальнейшем использовать.

#### Генерация случайных чисел

Класс **Random** в **C#** представляет собой генератор псевдослучайных чисел - т.е. данный класс отвечает за выдачу последовательности чисел, отвечающую определенным статистическим критериям случайности.  
  
**Random rnd = new Random();**   
  
Здесь мы объявили экземпляр класса **Random** (**rnd**), с помощью которого мы будем генерировать случайные числа. В дальнейшем мы будем использовать код, вида **rnd**.**Next** (диапазон\_генерации) или **rnd**.**Next** (от, до **-1**) для генерации случайного числа.   
  
Также мы объявим еще несколько переменных, часть из которых сразу будет инициализирована.

Point tmp\_location;

int \_w = System.Windows.Forms.SystemInformation.PrimaryMonitorSize.Width;

int \_h = System.Windows.Forms.SystemInformation.PrimaryMonitorSize.Height;

**tmp\_location** объявляется для того, чтобы в будущем временно хранить текущее положение окна.   
  
Также нам следует подумать о том, что при случайном перемещении, наше окно может выйти далеко за пределы экрана.

#### Чтобы определить разрешение экрана в **C# .NET**, мы будем использовать:

System.Windows.Forms.SystemInformation.PrimaryMonitorSize

**\_h** и **\_w** будут хранить в себе размеры экрана пользователя, которые определяются при их инициализации.   
Теперь код этой функции будет выглядеть следующим образом:

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

// переводим координату X в строку и записывает в поля ввода

textBox1.Text = e.X.ToString();

// переводим координату Y в строку и записывает в поля ввода

textBox2.Text = e.Y.ToString();

// если координата по оси X и по оси Y лежит в очерчиваемом вокруг кнопки "да, конечно" квадрате

if (e.X > 80 && e.X < 195 && e.Y > 100 && e.Y < 135)

{

// запоминаем текущее положение окна

tmp\_location = this.Location;

// генерируем перемещения по осям X и Y и прибавляем их к хранимому значению текущего положения окна

// числа генерируются в диапазоне от -100 до 100.

tmp\_location.X += rnd.Next(-100, 100);

tmp\_location.Y += rnd.Next(-100, 100);

// если окно вылезло за пределы экрана по одной из осей

if (tmp\_location.X < 0 || tmp\_location.X > (\_w - this.Width / 2) || tmp\_location.Y < 0 || tmp\_location.Y > (\_h - this.Height / 2))

{ // новыми координатами станет центр окна

tmp\_location.X = \_w / 2;

tmp\_location.Y = \_h / 2;

}

// обновляем положение окна, на новое сгенерированное

this.Location = tmp\_location;

}

}

Вот, собственно, и все. Откомпеллировав приложение, можете попробовать нажать на кнопку "Да, конечно". Это будет крайне трудно.

Начало формы

* Используйте вашу учетную запись на Facebook.com для входа на сайт.
* Используйте вашу учетную запись VKontakte для входа на сайт.
* Используйте вашу учетную запись Google для входа на сайт.

Конец формы

Авторизуйтесь с помощью соц. сети или с помощью аккаунта на сайте:

Начало формы





Запомнить меня

Войтиhttp://esate.ru/core/images/preload.gif

Забыли пароль? Регистрация

Конец формы

**×**