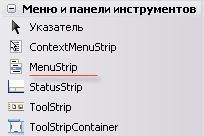
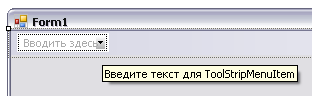
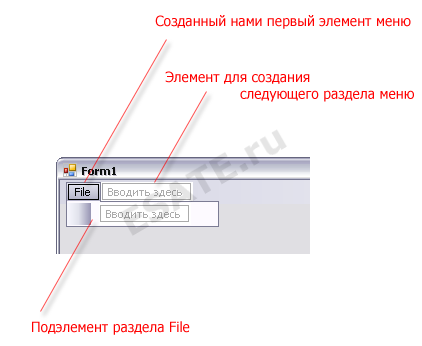
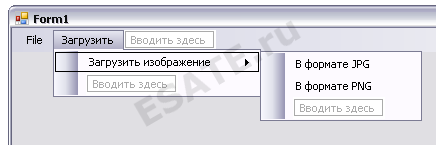
# Создание второго оконного приложения. (Создание меню, taskbar, дополнительных диалоговых окон, открытие и загрузка изображений).

Первоначально, создайте новый проект, указав в качестве шаблона **Windows Form Application** и назовите **second\_application**. Затем нажмите **OK**.

#### Добавление меню в приложение C# .NET

Здесь все предельно просто: чтобы добавить меню в приложение, сначала откройте окно **Toolbox** (Панель элементов).  
Нам потребуется разворот **Menus & Toolbars** (Меню и панели инструментов). В нем нас интересует элемент **MenuStrip** (рис. 1).  
Рисунок 1. Выбор элемента для созданию меню.   
Зажмите его левой клавишей мыши и перетащите на форму. На нашей форме отразиться элемент меню, как показано на рисунке 2.  
Рисунок 2. Создание элементов меню.   
Теперь, щелкнув на строке «*введите здесь*» вы можете ввести название меню: назовем его «**File**». При этом добавятся 2 дополнительных элемента – один снизу, он будет доступен при открытии меню, второй справа - для создания новых разделов меню (рис. 3).  
Рисунок 3. Процесс добавления элементов меню.

Дайте название подэлементу меню **File** “**Выход**”, как показано на рисунке 4. После этого назовите еще один раздел меню – «**загрузить**». В меню загрузить добавьте подэлемент «**загрузить изображение**» и в раскрывающемся меню назначьте ему 2 подэлемента: «**В формате JPG**» и «**В формате PNG**». Выглядеть меню будет следующим образом (рис. 4):  
Рисунок 4. Заполнение разделов создаваемого меню.   
Теперь рассмотрим то, как просто добавить обработчик события меню. Для этого перейдите к меню «**Выход**». Теперь сделайте двойной щелчок левой клавишей мыши – **MS Visual Studio** автоматически создаст код функции обработчика и настроит событие обработки.   
  
Перед нами откроется код функции обработчика:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

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

{

}

Так как этот элемент меню отвечает за выход из приложения, добавим в него код, который будет генерировать **MessageBox** с вопросом о подтверждении выхода из приложения. Если пользователь подтвердит выход – приложение будет завершено.   
  
Новый код функции с комментариями:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

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

{

// создаем переменную rsl, которая будет хранить результат вывода окна с вопросом

// (пользователь нажал одну из клавиш на окне - это и есть результат)

// MessageBox будет содержать вопрос, а также кнопки Yes, No и иконку Question (Вопрос)

DialogResult rsl = MessageBox.Show ("Вы действительно хотите выйти из приложения?", "Внимание!", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

// если пользователь нажал кнопку да

if (rsl == DialogResult.Yes)

{ // выходим из приложения

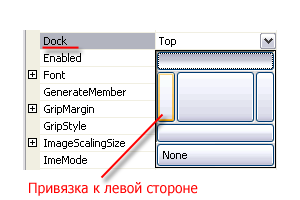
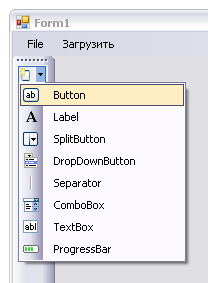
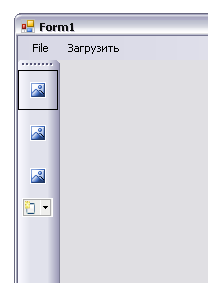
Application.Exit();

}

}

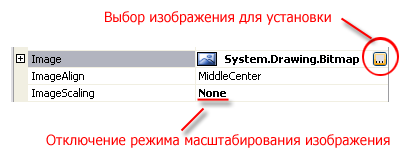
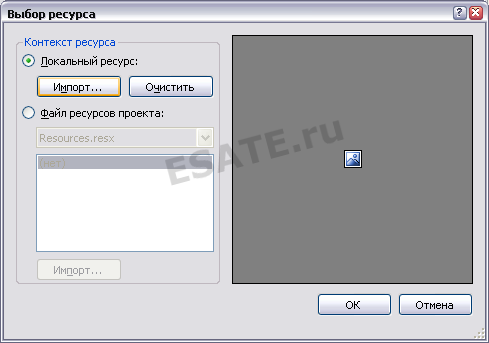
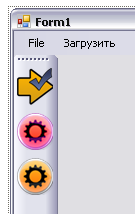
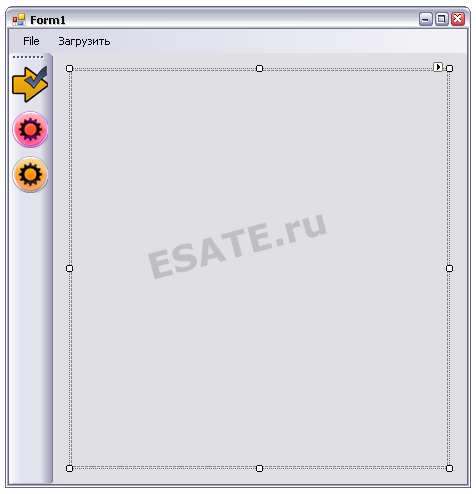
Теперь можно откомпилировать приложение и проверить работоспособность кнопки.   
  
Часто панель управления (**Toolbar**) дублирует элементы меню для быстрого к ним доступа.

#### Создание панели управления (Toolbar) в C# .NET

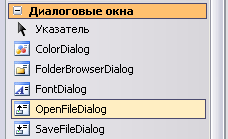
Нам снова нужно перейти к окну **Toolbox** (Панель инструментов) и развороту **Menus & Toolbars**. В этот раз мы выберем элемент **ToolStrip** (рис. 5).  
Рисунок 5. Окно Toolbox.   
Перетащите элемент управления на окно и вы увидите, как вдоль его верхней границы разместиться панель **ToolBar** (рис. 6).  
Рисунок 6. Добавленный элемент Toolbar.   
Мы изменим положение привязки нашего **ToolBar**’a. Для этого щелкнем по нему правой кнопкой и в открывшемся контекстном меню выберем пункт Свойства. Откроется окно свойств: здесь мы изменим привязку на левую часть окна, внеся изменения в параметр **Dock**, как показано на рисунке 7.  
Рисунок 7. Привязка панели инструментов к левой стороне.   
Теперь увеличим размеры кнопок на **Toolbar**'e. Для этого сначала необходимо в его свойствах установить параметр **AutoSize** равным **false**. Теперь мы можем изменить размеры самих кнопок: установим параметры **Size** - **Width** равным **44**. Поле станет шире (рис. 8).  
Уроки OpenGL + C#: Установка размеров изображений, размещаемых на кнопках создаваемого ToolbarРисунок 8. Установка размеров изображений, размещаемых на кнопках создаваемого Toolbar.   
Теперь добавим 3 кнопки на наш **ToolBar**. Для этого щелкните по нему и в раскрывающемся списке элементов, которые мы можем добавить, выберите элемент **button** (рис. 9).  
Рисунок 9. Добавление кнопок на панель элементов (Toolbar).   
Повторите операцию, чтобы кнопок на панели стало две. Теперь поочередно выберите каждую кнопку и в ее свойствах установите **AutoSize** равный **false**. После это перейдите к полю **Size** и установите высоту равную **42**. Теперь кнопки примут квадратный вид.   
  
Таким образом, на панели разместятся 3 кнопки, как показано на рисунке 10.  
Рисунок 10. Пример добавленных кнопок с установленными размерами.   
Теперь назначим изображения для данных картинок. В качестве изображений можно использовать все современные форматы, в том числе и **png24** с поддержкой прозрачности.   
  
Мы будем использовать 3 следующих изображения:

Первое  
Изображение для кнопки 1: будет назначено кнопке, отвечающей за открытие дополнительного диалогового окна.   
  
Два следующих изображения будут назначены кнопкам, которые будут отвечать за загрузку файлов **jpg** и **png**.

Второе  
Изображение для кнопки 2.   
  
Третье  
Изображение для кнопки 3.  
  
(!!!) **Обратите внимание**, что у данных изображений прозрачный фон.

2 кнопки будут дублировать меню с функциями загрузки изображений, 1-я кнопка будет предназначена для вызова окна, с отображением картинки, которую мы загрузили.   
  
Теперь для установки изображений необходимо перейти в свойства картинки, после чего мы установим значение параметра **ImageScaling** равным **none**, чтобы изображение не масштабировалось. Теперь в параметре **Image** мы можем выбрать изображение для загрузки, как показано на рисунке 11.  
Рисунок 11. Установка изображений на кнопки.   
В процессе выбора откроется окно, показанное на рисунке 12.  
Рисунок 12. Импорт рисунка для установки на кнопке.   
Теперь щелкните на кнопке **Import** и выберите необходимый рисунок. Аналогично повторите с другими рисунками. В результате вы получите 3 красивые кнопки, как показано на рисунке 13.  
Рисунок 13. Пример созданной панели инструментов.   
Для того чтобы создать обработчики нажатий на кнопки этого **ToolBox**'а, достаточно совершить двойной щелчок мыши на каждом из них – **MS Visual Studio** автоматически сгенерирует код обработчик события и заготовки функций.   
  
В будущем мы добавим вызов необходимых нам функций из этого обработчика.   
  
Теперь мы разместим на форме элемент **PictureBox** и настроим размеры окна, чтобы оно выглядело следующем образом (рис. 14).  
Рисунок 14. Расположение элемента PictureBox в окне программы.   
В свойствах добавленного элемента **PictureBox** установите параметр **SizeMode**, равный **StretchImage**. Теперь, когда мы реализуем загрузку изображения, оно будет масштабироваться под размеры нашего элемента **PictureBox**.

#### Создание окна выбора файла в C# .NET

Чтобы пользователь мог выбирать файл для загрузки через стандартное в **Windows** окно загрузки файлов, мы выполним следующие действия.   
Перейдите к окну **ToolBox** (Панель элементов).   
Теперь перетащите элемент управления **OpenFileDialog** (рис. 15) на форму.  
Рисунок 15. Добавление элемента OpenFileDialog.   
Местоположение, куда вы перетащите элемент, неважно: он добавится в поле под окном, к другим специфическим объектам (рис. 16).  
Уроки OpenGL + C#: Элемент OpenFileDialog, расположенный под редактируемой формойРисунок 16. Элемент OpenFileDialog, расположенный под редактируемой формой.   
Как видно из рисунка 16, к дополнительным элементам, уже установленным на наше окно (меню и **ToolBox**), добавился еще и элемент **OpenFileDialog1**.   
Теперь мы реализуем код открытия окна выбора файла, с необходимыми нам параметрами.   
  
Если вы еще не создали обработчики нажатия на кнопки загрузки и элементы меню, так же предназначенные для загрузки, сделайте это с помощью двойных щелчков по ним.   
Код сгенерированных функций выглядит следующим образом:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

// обработка кнопки меню "загрузка - в формате jpg"

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

{

}

// обработка кнопки меню "загрузка - в формате png"

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

{

}

// обработка кнопки №2 на панели

private void toolStripButton2\_Click(object sender, EventArgs e)

{

}

// обработка кнопки №3 на панели

private void toolStripButton3\_Click(object sender, EventArgs e)

{

}

Теперь напишем следующую функцию **LoadImage**.

#### Открытие окна выбора файла и загрузка изображения в C# .NET

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

Image MemForImage;

// функция загрузки изображения

private void LoadImage(bool jpg)

{

// директория, которая будет выбрана как начальная в окне для выбора файла

openFileDialog1.InitialDirectory = "c:";

// если будем выбирать jpg файлы

if(jpg)

{

// устанавливаем формат файлов для загрузки - jpg

openFileDialog1.Filter = "image (JPEG) files (\*.jpg)|\*.jpg|All files (\*.\*)|\*.\*";

}

else

{

// устанавливаем формат файлов для загрузки - png

openFileDialog1.Filter = "image (PNG) files (\*.png)|\*.png|All files (\*.\*)|\*.\*";

}

// если открытие окна выбора файла завершилось выбором файла и нажатием кнопки ОК

if(openFileDialog1.ShowDialog() == DialogResult.OK)

{

try // безопасная попытка

{

// пытаемся загрузить файл с именем openFileDialog1.FileName - выбранный пользователем файл.

MemForImage = Image.FromFile(openFileDialog1.FileName);

// устанавливаем картинку в поле элемента PictureBox

pictureBox1.Image = MemForImage;

}

catch (Exception ex) // если попытка загрузки не удалась

{

// выводим сообщение с причиной ошибки

MessageBox.Show( "Не удалось загрузить файл: " + ex.Message);

}

}

}

Функция **LoadImage** в качестве входного параметра будет получать флаг о том, какой фильтр для выбора файлов необходимо выбрать. Далее вызывается окно выбора файла и если оно при закрытии возвращает результат **DialogResult.OK**, то мы пытаемся загрузить и установить выбранную картинку в поле **PictureBox**. Конструкция **try catch** необходима нам здесь по следующей причине: если загрузка прошла неудачно, то мы получим сообщение об ошибке, но на выполнение программы это не повлияет, и мы сможем продолжить ее выполнение.   
  
Далее функции обработчики нажатий пунктов меню и кнопок на панели управления реализуют вызов функции загрузки с необходимым флагом.

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

// обработка кнопки меню "загрузка - в формате jpg"

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

{

// загружаем jpg-файлы

LoadImage(true);

}

// обработка кнопки меню "загрузка - в формате png"

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

{ // загружаем png файлы

LoadImage(false);

}

// обработка кнопки №2 на панели

private void toolStripButton2\_Click(object sender, EventArgs e)

{ // загружаем jpg фалйы

LoadImage(true);

}

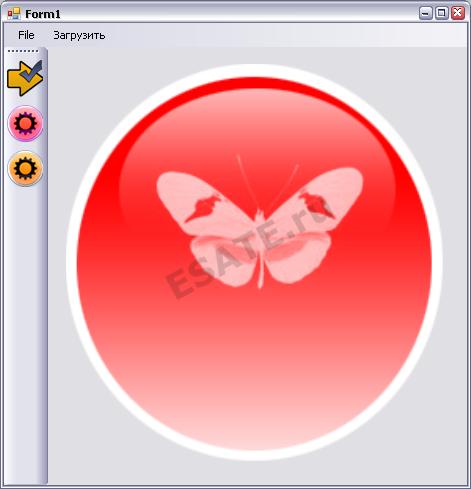
// обработка кнопки №3 на панели

private void toolStripButton3\_Click(object sender, EventArgs e)

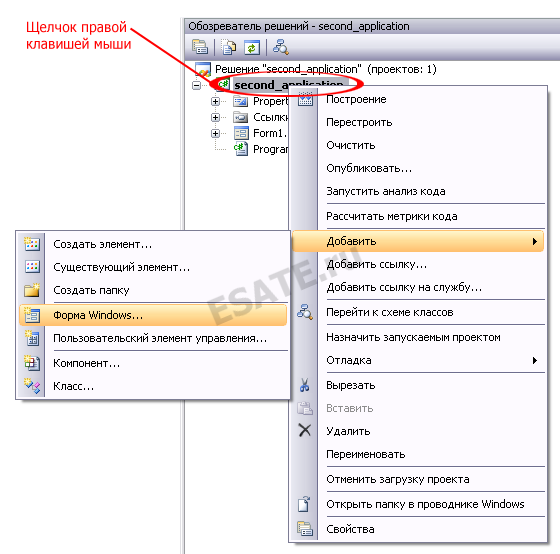
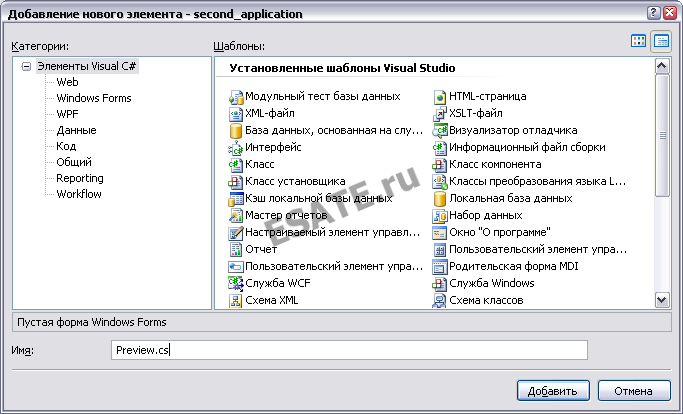
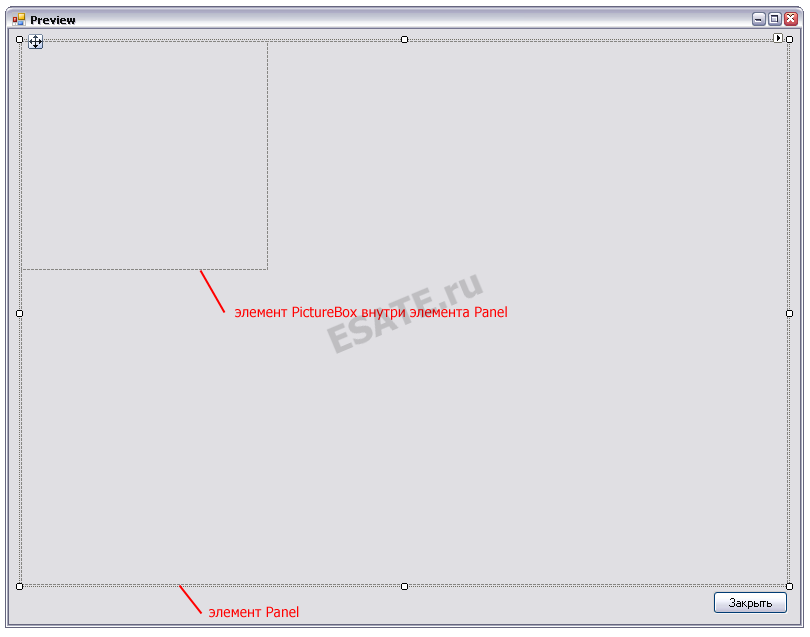
{ // загружаем png файлы

LoadImage(false);

}

Теперь, если откомпилировать приложение (**F5**), то можно попробовать загрузить изображение. Обратите внимание, что загрузка **png24** с альфа-каналом тоже работает (рис. 17).  
Рисунок 17. Пример загруженного изображения.   
Как видите, наше изображение масштабируется под размер элемента **PictureBox**. Поэтому сейчас мы добавим в проект еще одну форму, на которой мы будем отображать изображение в его истинном размере.

#### Добавление и вызов дополнительного диалогового окна

Для этого перейдите к окну **Solution** **Explorer** (Обозреватель решений), после чего щелкните на названии проекта правой кнопкой мыши и в открывшемся контекстном меню выберите **Add->Form**, как показано на рисунке 18.  
Рисунок 18. Добавление в проект нового диалогового окна.   
В открывшемся окне введите названия класса, который будет отвечать за генерацию нашего вспомогательного окна – **Preview.cs** (рис. 19).  
Рисунок 19. Установка названия создаваемой формы для нового диалогового окна.   
Теперь измените размер окна так, как показано на рисунке 20. Затем на форме разместите элемент **panel**. Внутри элемента **panel** разместите элемент **pictureBox**, как показано на рисунке 20.  
Рисунок 20. Размещение элементов panel и picturebox на созданном диалоговом окне.   
Данное окно будет получать при загрузке в качестве параметра ссылку на наше загруженное изображение. Затем оно будет устанавливать границы элемента **pictureBox** равными размерам полученного изображения, после чего устанавливать его в элемент **pictureBox**.   
  
Перейдите к свойствам элемента **panel** и установите значение параметра **AutoScroll** равное **true**. Теперь, в том случае, если размер изображения будет превышать размер элемента **panel1**, будут появляться вертикальная и горизонтальная полоса прокрутки, с помощью которых можно будет просмотреть изображение.   
  
Теперь рассмотрим исходный код, необходимый для реализации данных возможностей.   
  
Первым делом назначим обработчик кнопки с изображением стрелки направленной вправо на нашем главном окне. Щелкните по ней дважды, после чего вы перейдете к редактированию кода функции, которая будет вызвана при щелчке по данной кнопке.   
  
Он будет выглядеть следующим образом:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

// кнопка активации дополнительного диалогового окна

private void toolStripButton1\_Click(object sender, EventArgs e)

{

// создаем новый экземпляр класса Preview,

// отвечающего за работу с нашей дополнительной формой

// в качестве параметра мы передаем наше загруженное изображение

Form PreView = new Preview(MemForImage);

// затем мы вызываем диалоговое окно

PreView.ShowDialog();

}

Как видно из исходного кода, класс **Preview** получает в качестве параметра наше загруженное изображение. Для правильной работы программы мы должны изменить конструктор класса **Preview**, а также реализовать остальную функциональность программы.   
  
Перейдите к окну **Preview**, после чего сделайте двойной щелчок левой клавшей мыши на нем (**НЕ** на размещенных на нем объектах).   
  
Откроется для редактирования функция:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

private void Preview\_Load(object sender, EventArgs e)

Но мы сначала изменим код конструктора класса, теперь он будет выглядеть следующим образом:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

public partial class Preview : Form

{

// объект Image для хранения изображения

Image ToView;

// модифицируем конструктор окна таким образом, чтобы он получал

// в качестве параметра изображение для отображения

public Preview(Image view)

{

// получаем изображение

ToView = view;

InitializeComponent();

}

Теперь отредактируем код функции **Preview\_Load**. Он будет выглядеть следующим образом:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

// эта функция выполнится при загрузке окна

private void Preview\_Load(object sender, EventArgs e)

{

// если объект, хранящий изображение не равен null

if (ToView != null)

{

// устанавливаем новые размеры элемента pictureBox1,

// равные ширине (ToView.Width) и высоте (ToView.Height) загружаемого изображения.

pictureBox1.Size = new Size(ToView.Width, ToView.Height);

// устанавливаем изображение для отображения в элементе pictureBox1

pictureBox1.Image = ToView;

}

}

В заключении, самостоятельно добавьте обработчик нажатия кнопки закрыть. Функция закрытия будет выглядеть следующим образом:

/\*[http://esate.ru](http://esate.ru/), Anvi\*/

// обработчик кнопки закрыть

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

{

// закрываем диалоговое окно

Close();

}

Вот и все. Теперь, если будет загружено большое изображение, его отображение в дополнительном окне будет снабжено полосами прокрутки (рис. 21).  
Рисунок 21. Изображение с полосами прокрутки.   
Откомпилируйте приложение (**F5**), чтобы проверить его.

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

**×**