Окна и сообщения.

Отвагин Алексей Владимирович,
доцент каф. ЭВМ, к.т.н., а. 505-5

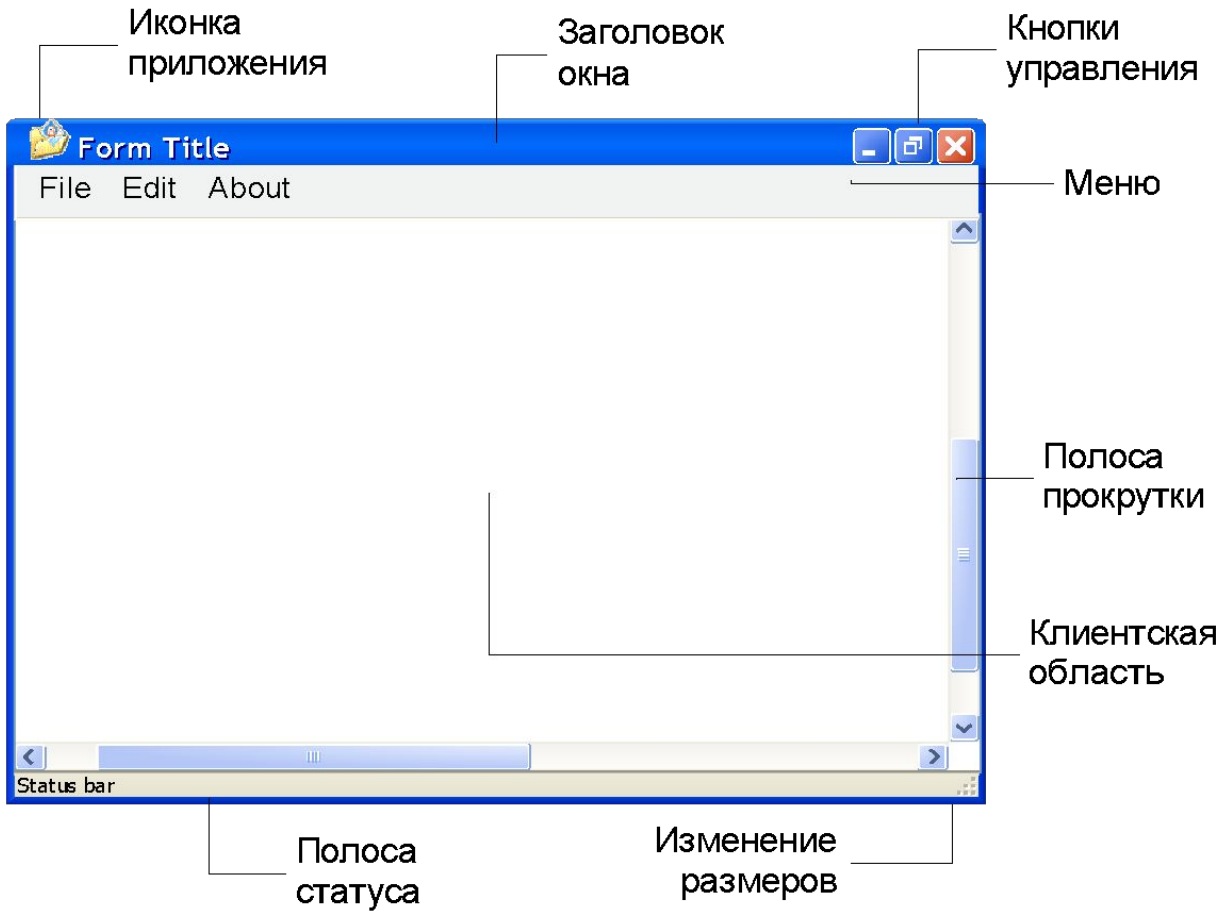
Содержание

- Понятие окна, разновидности окон
- Характеристики окон
- Сообщения, иерархия сообщений
- Механизм обработки сообщений

Понятие окна

- Приложения пользователя и элементы управления отображаются в интерфейсе через окна
- Окно – (прямоугольная) область экрана, предназначенная для сбора и/или вывода информации от пользователя
- Окно связано с определенным классом, реализующим его логику поведения
- Окна идентифицируются и управляются менеджером окон

Элементы окна



Разновидности окон

- Окна приложений
- Элементы управления
- Диалоги
- Окна многодокументного интерфейса

Создание окна

```
using System;
using System.Windows.Forms;
class MyFirstApp {
static void Main() {
 Form form = new Form();
    Application.Run(form);
 }
}
```

Характеристики окна

- Название окна
- Размер и позиция окна на экране
- Оформление окна (стиль рамки, кнопки, цветовые характеристики)
- Элементы управления
- Иконка, шрифт, курсор

Иерархия окон

- Окна располагаются на экране иерархически
- Позиция окна в иерархии определяется z-порядком
- Окна потомков всегда находятся перед окном родителя
- Сиблинги – окна, созданные одним родителем

Окно - контейнер

```
public Button button1;
 public Form1() {
  button1 = new Button();
  button1.Size = new Size(40, 40);
  button1.Location = new Point(30, 30);
  button1.Text = "Click me";
  this.Controls.Add(button1);
}
```

Сообщения

- Способ реакции интерфейса на происходящие вне и внутри его события
- Обработка сообщений возлагается на оконную процедуру
- Передача сообщений, их создание и управление очередью возлагается на механизмы интерфейса

Делегаты Windows

- Классы реализации механизма обработки событий
- Содержат ссылку на метод
- Делегат принимает ссылки, соответствующие его синтаксису
- Поддерживает мультикаст

```
delegate void EventHandler(object sender,
    EventArgs e);
```

Добавление обработчика сообщений

```
private void button1_Click(object sender,
    System.EventArgs e)
{
}
button1.Click += new
    EventHandler(button1_Click);
```

Иерархия сообщений

- System.EventArgs
  - System.Windows.Forms.PaintEventArgs
  - System.Windows.Forms.ControlEventArgs
  - System.Windows.Forms.MouseEventArgs
  - System.Windows.Forms.DragEventArgs
  - System.Windows.Forms.FormClosedEventArgs
  - System.Windows.Forms.KeyEventArgs
  - System.Windows.Forms.ScrollEventArgs
  - System.Drawing.Printing.PrintPageEventArgs
  - System.Windows.Forms.Integration.ChildChangedEventArgs

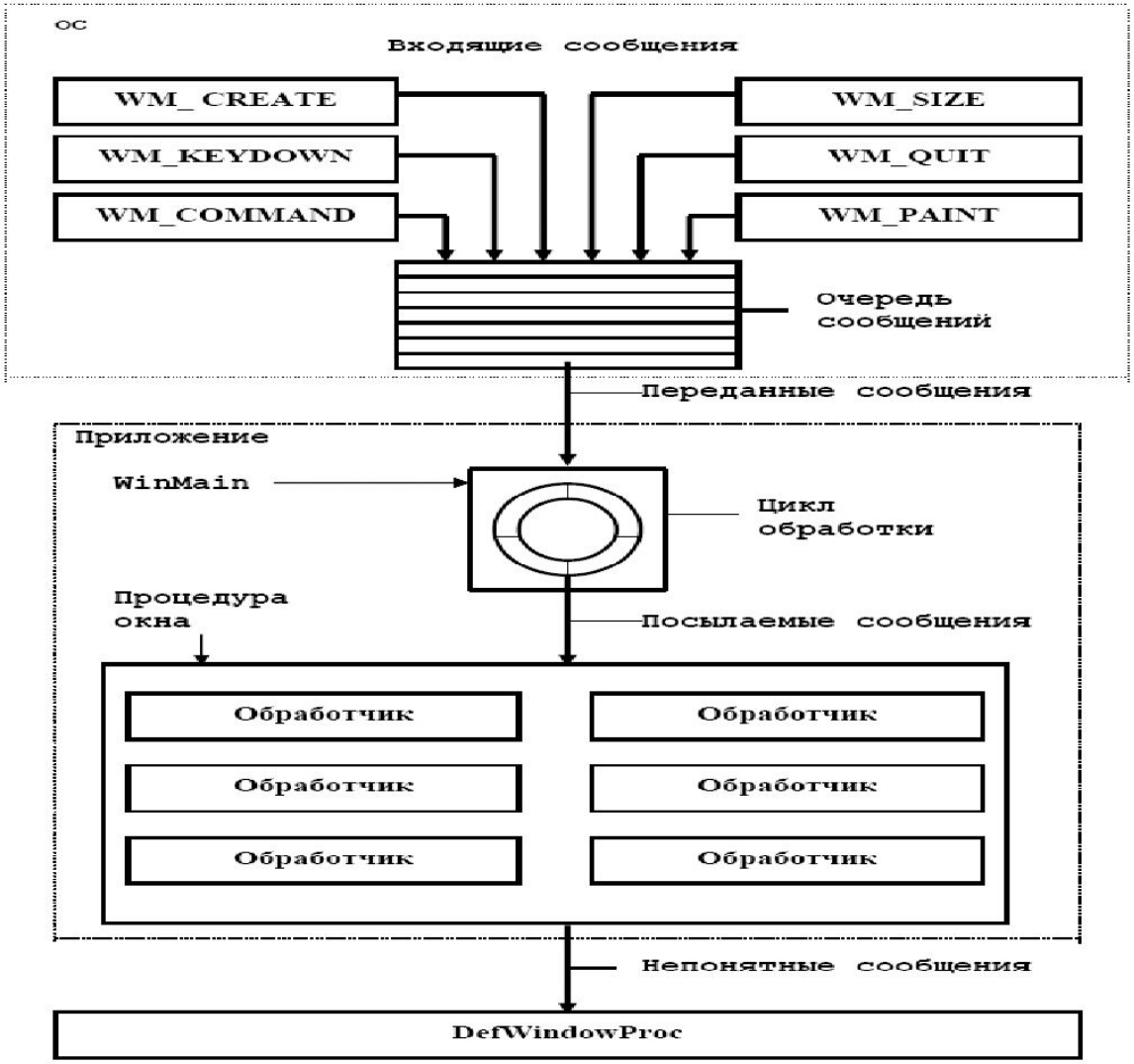
Создание событий на базе EventArgs

```
public class AlarmEventArgs : EventArgs
 {
  private readonly int nrings = 0;
  private readonly bool snoozePressed = false;
  //Constructor.
  public AlarmEventArgs(bool snoozePressed, int nrings)
  {
     this.snoozePressed = snoozePressed;
     this.nrings = nrings;
  }
 //Properties.
  public string AlarmText { ... }
  public int NumRings { ... }
  public bool SnoozePressed{ ... }
 }
```

Реализация события

```
public delegate void AlarmEventHandler(object sender, AlarmEventArgs e);
public class AlarmClock {
...
 public event AlarmEventHandler Alarm;
 protected virtual void OnAlarm(AlarmEventArgs e)
 {
    if (Alarm != null) Alarm(this, e);
 }
}
public class AlarmClock {
 ...
 public void Start() {
  ...
  System.Threading.Thread.Sleep(300);
  AlarmEventArgs e = new AlarmEventArgs(false, 0);
  OnAlarm(e);
 }
}
```

Обработка сообщений



Дескриптор сообщения

```
typedef struct {
 HWND hwnd;
 UINT message;
 WPARAM wParam;
 LPARAM lParam;
 DWORD time;
 POINT pt;
 } MSG, *PMSG;
```

дескриптор окна
тип сообщения
первый параметр
второй параметр
момент времени
координаты мыши

Сообщения в очереди

- Помещаются в системную очередь сообщений
- Вызываются событиями интерфейса, например, пользовательским вводом
- Функции для работы с сообщениями в очереди:
  - PostMessage(), PostThreadMessage()
  - GetMessage(), PeekMessage(), DispatchMessage()
  - GetMessageTime(), GetMessagePos()
  - WaitMessage()
  - SendMessageExtraInfo(), GetMessageExtraInfo()

Сообщения вне очереди

- Направляются непосредственно оконной процедуре
- Вызываются системой, например, событие создания окна, получение фокуса и т.д.
- Функции для работы с сообщениями вне очереди:
  - SendMessage(), SendMessageCallback()
  - BroadcastSystemMessage(), BroadcastSystemMessageEx()
  - SendMessageTimeout()

Основной цикл обработки сообщений

```
MSG msg;
BOOL bRet;
while( (bRet = GetMessage(&msg, NULL, 0, 0 )) != 0) {
   if (bRet == -1) {
   // ошибка
   }
else {
 TranslateMessage(&msg);
 DispatchMessage(&msg);
 }
}
```

- Функция `GetMessage()` возвращает FALSE, когда из очереди поступает WM_QUIT
- `TranslateMessage (&msg)` передает структуру *msg* обратно Windows для трансляции клавиатуры

Оконная процедура

- Обрабатывает все сообщения, поступающие в окно от устройств ввода или системы

```
switch (iMsg) {
    case WM_CREATE : [обработка WM_CREATE]
        return 0 ;
    case WM_PAINT : [обработка WM_PAINT]
        return 0 ;
    case WM_DESTROY : [обработка WM_DESTROY]
        return 0 ;
}
return DefWindowProc (hwnd, iMsg, wParam, lParam);
```

Порядок событий в окне

- Form.Load – загрузка формы
- Form.Activated – активация формы
- Form.Shown – первый вывод формы на экран

- Form.FormClosing – закрытие формы
- Form.FormClosed – форма закрыта
- Form.Deactivate – деактивация формы

События, связанные с фокусом

- Control.Enter – получение фокуса
- Control.GotFocus – получение фокуса
- Control.Leave – потеря фокуса
- Control.Validating – проверка данных
- Control.Validated – завершение проверки
- Control.LostFocus – потеря фокуса

Диалоговые окна

- Временные окна, создаваемые с целью приема пользовательского ввода
- Строятся на основе шаблона окна – перечня элементов управления с информацией об их размещении
- Предопределенные окна –диалоги для выполнения часто используемых операций, общих для всех приложений
- Диалоги жестко привязаны к владельцу, находясь в z-порядке над ним

Виды диалоговых окон

- Модальное – требует ввода информации или отмены окна, прежде чем приложение продолжит работу.

  ```
  Dialog.ShowDialog()
  ```

- Немодальное – позволяет вводить информацию и переключаться на основное окно приложения без закрытия.

  ```
  Dialog.Show()
  ```

- Модальные окна более просты в управлении

Вывод диалогового окна

```
private void
  button1_Click(object sender,
  System.EventArgs e) {
Form dlg1 = new Form();
dlg1.ShowDialog();
}
```

Получение данных через свойства

- Класс свойства, связанный с элементом управления

```
public string Name {
 get {
  return nameTextBox.Text;
 }
 set {
  nameTextBox.Text = value;
 }
}
```

Получение данных в диалоге

- Вывод диалога

```
private void ShowMyDialog() {
   Form1 dlg = new Form1();
   dlg.ShowDialog();
   if (dlg.DialogResult ==
   DialogResult.OK) {
   MessageBox.Show (dlg.Name);
  }
 }
```

Обработка кнопок диалога

```
void okButton_Click(object sender, EventArgs e) {
    this.DialogResult = DialogResult.OK;
    this.Close();
}
void cancelButton_Click(object sender, EventArgs e)
    {
    this.DialogResult = DialogResult.Cancel;
    this.Close();
}
```