ТЕМА: Событийно управляемый интерфейс

Цели:

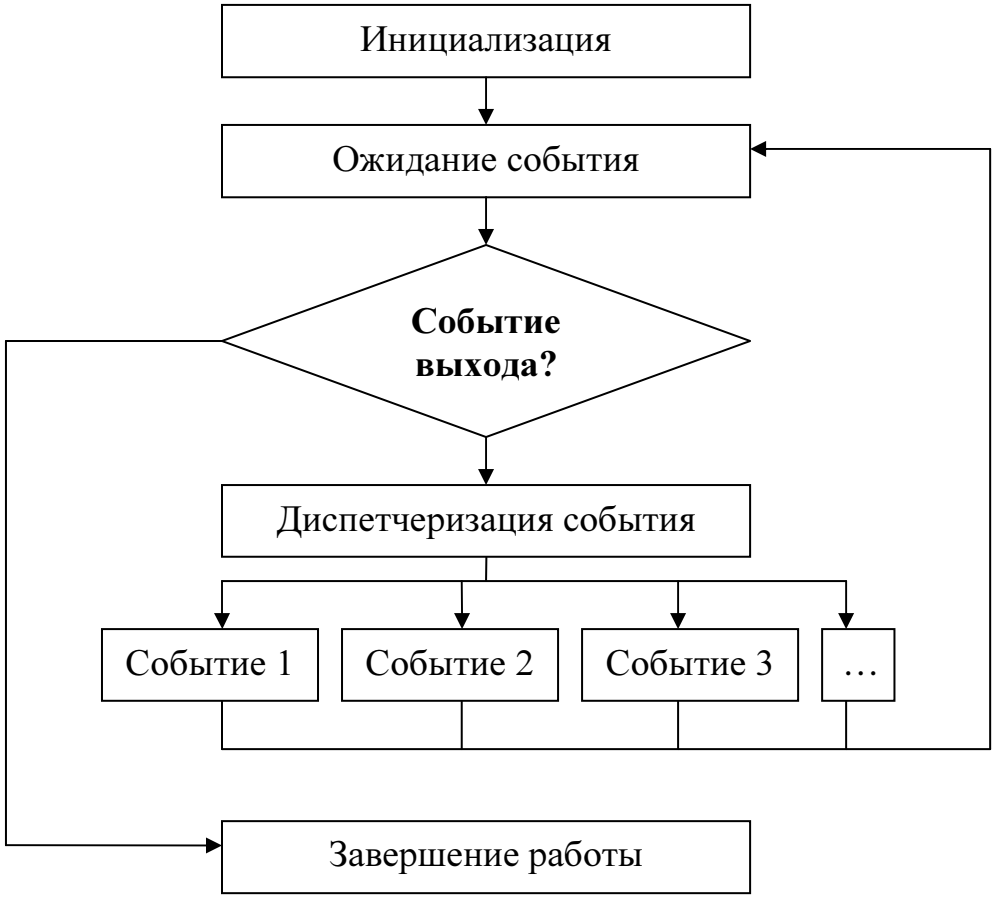
- Сформировать понятие о событийно управляемом интерфейсе
- Рассмотреть механизмы работы событийно управляемого интерфейса
- Познакомится с понятиями сообщения, события, очереди сообщений
- Рассмотреть механизмы построения событийно управляемого интерфейса на Lazarus в рамках LCL
- Рассмотреть основные события компонент LCL.
- Закрепить полученные знания построением простого приложения на Lazarus

Понятие событийно управляемого интерфейса

Для решения задачи управления окнами и множеством других объектов программы, в системе **Windows** используется механизм обработки событий.

Под ***событием*** понимается факт свершения элементарного действия, от которого может зависеть ход выполнения программы. Событиями, например, являются: нажатие клавиш на клавиатуре, перемещение мыши, истечение заданного промежутка времени и т.п.

Общий алгоритм программы, ориентированной на обработку событий, представлен на рисунке.



Сразу после инициализации программы управление передается на цикл обработки событий, который автоматически выполняет ряд операций:

- получение события
- проверку условия выхода из программы
- диспетчеризация события (передача события соответствующей процедуре обработки).

При поступлении события, указывающего на необходимость завершить программу, происходит выход из цикла обработки событий, и управление передается процедуре завершения работы программы, после чего осуществляется выход из нее.

Достоинство такой схемы управления в ее универсальности: чтобы в существующую программу добавить реакцию на новое событие, не нужно менять существующий исходный код. Достаточно определить новую процедуру, которая при возникновении этого события будет осуществлять необходимые действия.

События в системе Windows характеризуется:

- получателем (как правило окно, которому кому адресовано событие);
- типом, который связан с источником события;
- временем возникновения;
- положением на экране курсора мыши в момент возникновения события;
- в зависимости от типа, событие может характеризоваться набором дополнительных параметров, интерпретация которых специфична для каждого конкретного случая.

Организация событийно управляемого интерфейса в Windows

Операционная система Windows является многозадачной средой для выполнения программ, имеющей событийный интерфейс. При работе в такой среде программа взаимодействует с другими программами и операционной системой посредством сообщений.

Сообщения Windows

Все события, возникающие в среде Windows, приводятся к одному общему виду, называемому сообщением.

Сообщение - это запись определенной структуры, которая содержит исчерпывающую информацию о происшедшем событии.

Например в Lazarus она определена следующим образом:

```
TMsg = record
  hwnd :HWnd;
  message: Integer;
  wParam :Integer;
  lParam :Longint;
```

```
    time :Longint;
    pt :TPoint;
  end;
```

- Поле **hwnd** содержит 16-разрядный дескриптор окна, в котором возникло сообщение.
- В поле **message** помещается двухбайтный код (тип) сообщения, выделяющий данное сообщение из большого количества имеющихся в Windows, а так же сообщений, созданных самими программами.
- Поля **wParam** и **lParam** содержат дополнительную информацию и зависят от типа сообщения. В сообщениях, определяемых пользователем, они могут использоваться для передачи необходимой информации.
- В поле **time** система Windows помещает время в миллисекундах, которое истекло с момента запуска системы до постановки сообщения в очередь.
- Поле **pt** указывает позицию курсора мыши в экранных координатах на момент возникновения события.

Сообщения могут поступать в программу из различных источников, например:

- пользователь генерирует сообщения, нажимая клавиши на клавиатуре или перемещая мышь и нажимая ее кнопки;
- сама среда Windows может посылать сообщения прикладной программе для уведомления о тех или иных событиях;
- программа может вызвать функции Windows, результатом которых является посылка сообщений программе;
- прикладная программа может посылать сообщение самой себе;
- прикладная программа может посылать сообщения другим прикладным программам;

Столь большое количество возможных сообщений требует наличие стройной системы их обработки.

Очереди сообщений

В системе Windows есть очередь сообщений и при получении нового сообщения, Windows ставит его в очередь на обработку.

Очередь сообщений обрабатывается последовательно, соответственно адресат получает сообщения в том порядке, в котором они ему были посланы.

Существует системная очередь сообщений и своя очередь сообщений для каждого приложения и, иногда, потока обработки. Т.е. для каждого оконного приложения Windows организует отдельную очередь сообщений прикладной программы, и сообщения из системной очереди распределяются по приложениям.

В процессе распределения сообщений по приложениям Windows извлекает очередное сообщение из системной очереди, определяет, с каким оно окном связано, и помещает в очередь того приложения, которому принадлежит окно.

Обработку прикладной очереди сообщений осуществляет уже само приложение. Для этого программа организует так называемый цикл обработки сообщений. В нем осуществляется выбор нового сообщения из очереди прикладной программы и вызов диспетчера для его обработки соответствующей функцией. В связи с тем, что сообщения связаны с окнами, функции обработки сообщения называют оконными функциями.

Событийно управляемый интерфейс и LCL

При разработки приложений в Дельфи, программисту нет необходимости создавать свой диспетчер обработки сообщений приложения и следить за распределением сообщений между окнами приложения. Классы LCL производят все эти действия автоматически.

Объект ***Application***, представляющий собой приложение, содержит диспетчер очереди сообщений. Для принудительного вызова диспетчера сообщений у **TApllication** есть метод ***ProcessMessages***. Данный метод позволяет обработать все сообщения, находящиеся на данный момент в очереди приложения. Это полезно, например, когда программа выполняет длительный расчет и по ходу расчета необходимо отображать его текужее состояние. В этом случае, чтобы окна программы не ‘залипали’ и информация на них менялась, необходимо отрабатывать сообщения перерисовки окон с помощью ***ProcessMessages***.

Класс ***TForm*** – основной класс для создания форм, также автоматически обрабатывает ряд сообщений, полученных от объекта ***Application***. Например, автоматически обрабатываются сообщения об изменении размеров окна, изменении его положения, записи данных в поля ввода, перерисовка формы и т.п.

Для программного управления событийным интерфейсом, в каждом компоненте LCL предусмотрены ***обработчики событий***. Они представляют собой процедуры, автоматически вызывающиеся компонентом по приходу ему соответствующего сообщения. Таким образом, каждый компонент LCL содержит ряд встроенных шаблонов для создания процедур-обработчиков.

Список событий, доступных для текущего компонента можно посмотреть в разделе **Events** справки. Для визуальных компонентов их список доступен в инспекторе объектов (Object Inspector) на вкладке **Events**.

Основные события компонентов LCL

Каждый обработчик событий получает ряд параметров, характеризующих сообщение. Набор параметров является специфическим

для каждого сообщения, однако любой обработчик получает в качестве параметра объект ***Sender*** типа **TObject**. Этим параметром передается объект, который был инициатором обработчика сообщения (т.е. тем компонентом, который вызвал обработчик).

Использование ***Sender*** позволяет задать один и тот же обработчик сообщения для нескольких компонент, осуществив преобразование типа переменной ***Sender*** внутри обработчика.

Рассмотрим основные обработчики событий компонент LCL.

События визуальных компонент

У каждого из визуальных компонент LCL есть общие события, связанные с интерфейсом пользователя, а именно – события обработки клавиатуры и мыши.

- ***OnEnter*** – событие, возникающее когда фокус ввода перемещается на компонент.
- ***OnExit*** – событие, возникающее когда компонент теряет фокус ввода.
- ***OnMouseDown*** – событие, возникающее когда пользователь нажимает (не отпуская) кнопку мышки над компонентом.
- ***OnMouseUp*** – событие, возникающее когда пользователь отпускает кнопку мышки над компонентом.
- ***OnMouseMove*** – событие, возникающее когда пользователь перемещает указатель мыши над компонентом.
- ***OnKeyDown*** – событие, возникающее когда пользователь производит нажатие клавиши на клавиатуре. Данное событие получает в качестве аргумента расширенный код нажатой клавиши и состояние клавиш Shift, Ctrl, Alt и клавиш мыши.
- ***OnKeyUp*** – событие, возникающее когда пользователь отпускает нажатую на клавиатуре клавишу. Данное событие получает аналогичные параметры, что и ***OnKeyDown***, однако здесь речь идет о коде отпущенной клавиши.
- ***OnKeyPress*** – событие, возникающее когда пользователь производит нажатие и отпускание клавиши клавиатуры. Данное событие отличается от ***OnKeyDown*** и ***OnKeyUp*** тем, что получает в качестве аргумента код нажатой клавиши и не может отследить состояние нажатия клавиш Shift, Ctrl и Alt, а так же кнопок мыши.
- ***OnResize*** – событие, возникающее когда пользователь или программа изменили размер компонента. Данное событие актуально для компонент-контейнеров, содержащих в себе другие компоненты. В обработчике этого события, например, можно изменять размеры других компонентов для более правильного их размещения внутри текущего.

TApplication

- ***OnMessage*** – событие, возникающее когда приложение получает сообщение Windows. Данный обработчик получает событие в том виде, в котором оно приходит от Windows. Входным аргументом у данного обработчика является запись типа **TMsg** (структура описана выше)**,** в которой содержится информация о сообщении. Второй параметр – ***Handled***, представляет булевское значение, устанавливаемое обработчиком в ***true***, если сообщение обработано и может быть извлечено из очереди сообщений приложения. В противном случае сообщение передается далее и обрабатывается встроенными обработчиками LCL.
- ***OnIdle*** – событие, возникающее когда приложение находится в режиме ожидания сообщений. Данный обработчик вызывается в случае, когда очередь сообщений приложения пуста и приложение ожидает события, не выполняя никаких действий.

TForm

- ***OnCreate*** – событие, возникающее после создания формы. Как правило, данное событие производит инициализацию переменных и компонент, размещенных на форме.
- ***OnClose*** – событие, возникающее после закрытия формы. В данном событии можно предусмотреть действия по деинициализации или сохранение каких-либо параметров.
- ***OnCanClose*** – событие, возникающее когда пользователь пытается закрыть форму. В качестве параметра в событие передается булевская переменная ***CanClose***, установка которому значения ***false*** предотвратит закрытие формы.
- ***OnCanResize*** – событие, возникающее когда пользователь меняет размер формы. Данное событие позволяет регулировать изменения размеров формы, например, для сохранения аспекта (соотношения размеров сторон формы). В качестве параметра в событие передается булевская переменная ***CanResize***, установка которому значения ***false*** предотвратит изменение размеров формы.

TEdit, TMemo

- ***OnChange*** – событие, возникающее когда пользователь вводит в компонент какие-либо данные, меняя тем самым поле ввода компонента. В процессе ввода событие может сработать несколько раз – по одному разу на каждое изменение содержимого компонента.

TButton

- ***OnClick*** – событие, возникающее когда пользователь нажимает на кнопку (посредством мыши или, если это возможно, клавиатуры). В данном обработчике содержатся действия, производимые данной кнопкой.

Задание:

Рассмотрите описанные выше события и определите, какие аргументы они получают.

Контрольные вопросы:

1. Что такое событийно управляемый интерфейс?
2. Как он организуется в Windows?
3. Что такое сообщение?
4. Что такое очередь сообщений?
5. Какие бывают очереди сообщений?
6. Как реализуется событийно-управляемый интерфейс в LCL?
7. Как обрабатываются сообщения в LCL?
8. Что такое событие?
9. Какие события вы знаете?
10. Возможно ли в LCL получить сообщение в чистом виде (как оно было в Windows)?