ТЕМА: Обработчики событий в LCL

Цели:

- Закрепить знания о механизмах построения событийно управляемого интерфейса на Deplhi в рамках LCL
- Научится задавать обработчики событий компонент LCL.
- Закрепить полученные знания построением простого приложения на Lazarus

Событийно управляемый интерфейс в 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.

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**.

В предлагаемой работе используются следующие события:

- ***TForm, OnCreate*** – событие, возникающее после создания формы. Как правило, данное событие производит инициализацию переменных и компонент, размещенных на форме. Событие возникает только один раз – при создании формы.
- ***TTimer, OnTimer*** – возникает при истечении интервала, заданного свойством ***Interval*** компонента **TTimer**. Интервал срабатывания события задается в миллисекундах, так значение 1000 соответствует одной секунде. Событие будет срабатывать только в том случае, если свойство ***Enabled*** таймера имеет значение ***true***, при значении ***false***, событие не происходит.
- ***TPanel, OnClick*** – событие, возникающее после нажатия и отпускания левой кнопки мышки (клика) над панелью. Обработчик данного события вызывается каждый раз, когда производится клик мышкой над компонентом.
- ***TShape, OnMouseDown*** – событие, возникающее после нажатия любой кнопки мыши над компонентом **TShape**. Компонент TShape не имеет события клика мышки (***OnClick***), поэтому отловить нажатие или отпускание кнопки мыши над компонентом можно с помощью событий ***OnMouseDown*** или ***OnMouseUp***, соответствующим нажатию и отпусканию кнопок мыши. В качестве аргументов, данным событиям

передается структура, содержащая информацию о том, какая именно кнопка нажата. Значение ***mbLeft*** соответствует левой кнопке мыши, ***mbRight*** – правой.

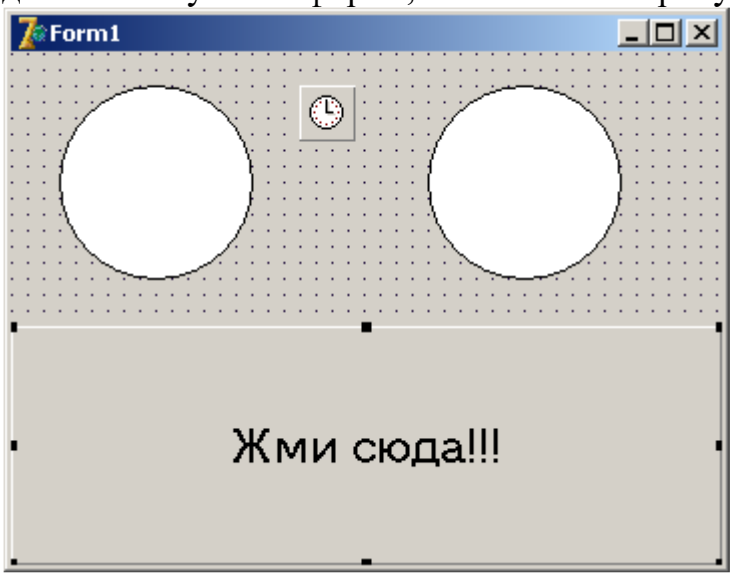
События компонента TTimer

Создадим новое приложение в Lazarus. Разместим на форме два компонента TShape (вкладка Additional), компонент TTimer (вкладка System) и компонент TPanel (вкладка Standard).

Зададим свойства объектов.

Компонент	Свойство	Значение
Timer1	Interval	500
Timer1	Enabled	false
Shape1	Shape	stCircle
Shape2	Shape	stCircle
Panel1	Align	alBottom
Panel1	Height	120
Panel1	Caption	Жми сюда!!!

В результате, должна получится форма, показанная на рисунке.



Определим для компонент формы обработчики событий. Начнем с события ***OnCreate*** компонента ***Form1***. Здесь мы зададим инициализационные значения свойств некоторых компонент.

Так, раскрасим ***Shape1*** и ***Shape2*** в зеленый и красный цвет соответственно. Компонент ***Shape2*** сделаем невидимым. Свойство ***Tag*** компонента ***Panel1*** установим в ноль.

```
begin
Shape1.Brush.Color:=clLime;
 Shape2.Brush.Color:=clRed;
 Shape2.Visible:=false;
 Panel1.Tag:=0;
end;
```

Определим событие ***OnClick*** компонента ***Panel1***. По этому событию будет запускаться и останавливаться таймер. Текущее состояние таймера определяется свойством ***Enabled***. Если таймер остановлен, панель возвращается в исходное состояние.

```
begin
 Timer1.Enabled:=not Timer1.Enabled;
 if not Timer1.Enabled then
  begin
   Panel1.Tag:=0;
   Panel1.Caption:='Жми сюда!!!';
  end;
end;
```

Определим событие ***OnTimer*** компоента ***Timer1***. По событию от таймера фигуры ***Shape1*** и ***Shape2*** будут по очереди появляться на экране.

```
begin
  Shape1.Visible:=not Shape1.Visible;
  Shape2.Visible:=not Shape2.Visible;
end;
```

Определим событие ***OnMouseDown*** для компонент ***Shape1*** и ***Shape2***. По нажатию на компонент будет проверяться, был ли он в этот момент видимым. Если да, то будет зарегистрировано попадание, иначе – промах.

Т.к. обработчик событий будет производить одни и те же действия, но для разных компонент – воспользуемся аргументом ***Sender***, который будет указывать на инициализировавший событие объект. Для преобразования ***Sender*** к нужному типу, воспользуемся конструкцией ***TShape(Sender)***.

При попадании в фигуру, зачитывается очко, которое отображается на панели. Через каждый 10 очков, производится ускорение таймера, путем уменьшения интервала срабатывания события ***OnTimer***.

```
begin
  if (Button=mbLeft) and TShape(Sender).Visible then
    begin
     Panel1.Tag:=Panel1.Tag+1;
     Panel1.Caption:='Î÷êîâ '+IntToStr(Panel1.Tag);
     if (Panel1.Tag mod 2=0) and (Timer1.Interval>10) then
      Timer1.Interval:=Timer1.Interval-10;
    end;
end;
```

ВАЖНО!!! Убедитесь, что данное событие задано для обоих фигур.

Попробуйте изменить событие OnClick на OnMouseDown, OnMoseDown на OnMouseUp и определить, что как этом измениться работа приложения.

Измените события таким образом, чтобы нажатие на правый компонент **TShape** производилось правой кнопкой мыши, а на левый – левой.

Добавьте еще один компонент **TShape** и реализуйте поочередное включение для трех компонент, с регистрацией нажатия на видимый из них.

Контрольные вопросы:

1. Как реализуется событийно-управляемый интерфейс в LCL?
2. Как обрабатываются сообщения в LCL?
3. Что такое обработчик событий?
4. Какие события вы рассмотрели?
5. Зачем нужно событие OnTimer? Где оно используется?
6. Зачем нужно событие OnClick? Где оно используется?
7. Зачем нужно событие OnMouseDown? Где оно используется?
8. Зачем нужно событие OnMouseUp? Где оно используется?
9. Как работает таймер?