

C++ part 5

Ned Lowe

ned.lowe@ic.ac.uk

Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning.

Events

- C++ Builder is event driven
- When an 'event' occurs, the program traps it and executes the programmed response
- Events include:
 - Buttons being clicked
 - Mouse actions
 - Time elapsing
 - many, many more

Properties

- *name->property* = “Value”;
- Initial properties can be set in ‘Object Inspector’
- Remember case-sensitive
- Set the ‘name’ attribute in the Object Inspector
- *label->Caption* = “Hello”;
- *label->Left* = 300;

Random Numbers

- `#include <stdlib.h>`
- `randomize();`
- `randomint = random(1000);`
- `Button1->Left = randomint;`

Creating a simple program

- Click an element, then draw it on the 'form'
- Click the 'Button' icon then draw it
- Change 'Caption' to 'Click Me!' in Object Inspector
- Click the 'Label' icon then draw it
- Change 'Name' to 'JumpLabel' in Object Inspector
- Change 'Caption' to 'Watch Me!' in Object Inspector

- Click the 'Play' button to compile, or press F9

Adding code

- Double-click Button, this creates the 'OnClick' event handler
- Type:

```
randomize();
```

```
Label1->Left = random(500);
```

```
Label1->Top = random(300);
```

```
within function start and end { }
```

Saving Text Files

- Add a 'Save Dialog' to the form
- Change 'DefaultExt' to 'txt'
- Click Filter, then ...
 - Filter Name = 'Text Files'
 - Filter = '*.txt'
- Add a Memo Component
- Add a Button
- Double-click Button and add:
if (SaveDialog1->Execute())
 Memo1->Lines->SaveToFile(SaveDialog1->FileName);

Explanation

- Save Dialog allows user to select where to save file
- Filter only displays *.txt files in the window
- DefaultExt saves file as *name.txt*
- Memo1->Lines->SaveToFile saves all the text in the memo component to the file specified

Loading Text Files

- Add a 'Open Dialog' to the form
- Click Filter, then ...
 - Filter Name = 'Text Files'
 - Filter = '*.txt'
- Add a second Button
- Double-click Button and add:

```
if (OpenDialog1->Execute())
```

```
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
```

On Screen Messages

- `MessageDlg("Message", type, TMsgDlgButtons() << button, 0);`
- **Type**
 - `mtWarning`
 - `mtError`
 - `mtInformation`
 - `mtConfirmation`
- **Button**
 - `mbYes`
 - `mbNo`
 - `mbOK`
 - `mbCancel`
- **Multiple Buttons** - `<< mbYes << mbNo << etc`

Message Return Values

- Each message can return a value based on what the user selected
- *mrbutton*
 - mrYes
 - mrNo
 - mrCancel

```
if (MessageDlg("Are you sure you wish to exit?", mtConfirmation,  
    TMsgDlgButtons() <<mbYes << mbNo, 0) == mrYes)
```

```
    UserSelected = true;
```

Exiting a program

- Click the form
- Click 'Events' in the Object Inspector
- Double-click OnCloseQuery
- If variable 'CanClose' is true then program will exit when user selects exit, if false program will not exit
- Add:

```
if (MessageDlg("Are you sure you wish to exit?", mtConfirmation,  
    TMsgDlgButtons() <<mbYes << mbNo, 0) == mrYes)
```

```
    CanClose = true;
```

```
else
```

```
    CanClose = false;
```

Adding a simple menu

- Click 'MainMenu' then add to the form
- Double-click the Menu item
- Type '&File' then press 'Enter'
 - creates a top-level menu item called File
- Click the sub-menu space
- Type E&xit
 - The ampersand (&) is for keyboard shortcuts
- Double-click the 'Exit' word to create an OnClick handler
- Add
 - Close();