

World", dan menjadikannya sebagai window paling atas (window utama). Setelah itu kita menginstansiasi objek app dengan kelas MyApp. Selanjutnya, kita memanggil metode MainLoop dari objek app. MainLoop merupakan jantung dari aplikasi di mana event-event diproses dan dikirim kembali ke window untuk diproses. MainLoop akan membuat aplikasi kita dijalankan sampai window-nya kita tutup.

Untuk aplikasi yang masih sangat sederhana seperti di atas, kita dapat menggunakan kelas wxPySimpleApp yang diturunkan dari kelas wxApp. Dengan menggunakan kelas wxPySimpleApp kode program akan lebih singkat. Untuk selanjutnya, kita akan menggunakan kelas wxPySimpleApp. Jika program di atas ditulis ulang, maka akan tampak sebagai berikut:

```
from wxPython.wx import *

app = wxPySimpleApp()
frame = wxFrame(None,-1, "Hello World ")
frame.Show(1)
app.MainLoop()
```

Cara kerja program ini sama dengan cara kerja program di atas, hasil kedua program akan tampak sama seperti Gambar 2.

Selanjutnya kita akan membahas kontrol-kontrol umum yang disediakan wxPython berupa kelas-kelas serta penanganan event kontrol-kontrol tersebut.

wxFrame()

Unit kontrol ini berupa sebuah window (jendela). Kontrol-kontrol yang kita buat akan kita tempatkan di dalam frame. Untuk membuat frame, hanya diperlukan perintah-perintah sederhana berikut:

```
...
frame = wxFrame(None,-1,"Halo
Dunia",wxPoint(20,30),wxSize(200,300))
frame.Show(True)
...
```

Pada kode di atas, kita menciptakan objek wxFrame dengan nama frame. Parameter None pada wxFrame bahwa frame tersebut tidak memiliki *parent*. Kita memberikan ID -1 pada objek frame, dan menempatkan frame tersebut pada posisi (20,30) dan ukuran 200x300 di layar. Posisi dan ukurannya dinyatakan dalam

satuan pixel, sehingga posisi dan ukuran tergantung pada resolusi layar.

wxMenu dan wxMenuBar()

Lazimnya aplikasi GUI pada umumnya mempunyai menu dan menubar. Adanya menu akan mempermudah user untuk berinteraksi dengan program.

Event handling:

EVT_MENU(kontrol,ID,fungsi)

➔ Event ini dipanggil ketika user memilih/mengklik menu tertentu.

Untuk membuat menu di program kita, hal pertama yang harus kita lakukan adalah membuat objek menu dengan wxMenu(), kemudian menambahkan item-item ke dalam menu tersebut, kemudian objek wxMenu() ditambahkan ke dalam objek wxMenuBar(). Selanjutnya, kita tinggal mengaktifkan menu yang kita buat.

```
...
menu = wxMenu()
menu.Append(101, "Teh Manis")
menu.Append(102, "Jus Jeruk")
menubar = wxMenuBar()
menubar.Append(menu, "Minuman")
#mengaktifkan menu yang kita buat
self.SetMenuBar(menubar)
...
```

Penggalan kode di atas menunjukkan bagaimana cara menambahkan menu dalam program Anda.

wxStatusBar()

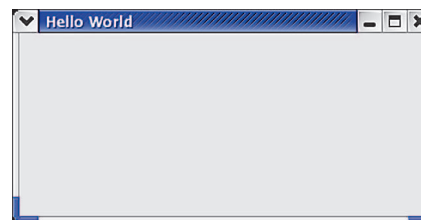
Status bar biasanya digunakan untuk menampilkan informasi atau status program yang kita jalankan.

Untuk membuat status bar ,Anda hanya perlu menambahkan beberapa baris perintah sebagai berikut:

```
...
#membuat status bar
frame.CreateStatusBar()
#mencetak teks tertentu pada status bar.
frame.SetStatusText("Ini contoh status bar")
...
```

wxMessageBox()

Untuk membuat kotak pesan, Anda dapat menggunakan wxMessageBox(). Berikut ini contohnya:



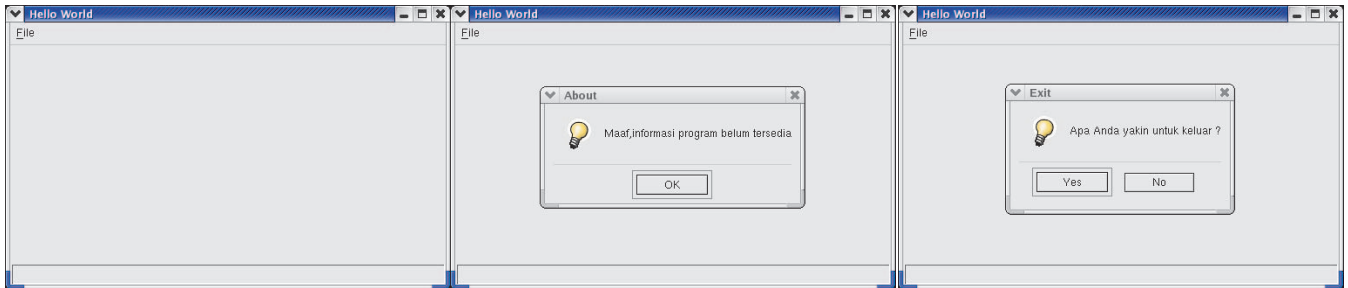
➤ Gambar 2. hello.

```
...
#membuat objek dlg yang mempunyai
tombol OK saja dan icon informasi.
dlg = wxMessageDialog(parent, "Ini contoh
message box","Message box",
wxOK|wxICON_INFORMATION)
#Tampilkan message box
dlg.ShowModal()
#Tutup message box ketika tombol OK
diklik atau diklik tanda silang pada
window sebelah kanan atas.
dlg.Destroy()
...
```

Berikut kita akan membuat contoh program yang mendemonstrasikan penggunaan ketiga kontrol di atas beserta penanganan event-nya :

```
from wxPython.wx import *

ID_ABOUT = 101
ID_EXIT = 102
class main_window(wxFrame):
    def __init__(self,parent,ID,title):
        wxFrame.__init__(self,parent,ID,title,
        wxDefaultPosition,size=(500,300),
        style=wxDEFAULT_FRAME_STYLE)
        #membuat status bar
        self.CreateStatusBar()
        menufile = wxMenu()
        menufile.Append(ID_ABOUT, "&About",
        "Informasi tentang program")
        menufile.AppendSeparator()
        menufile.Append(ID_EXIT, "E&xit","Keluar
dari program")
        EVT_MENU(self,ID_ABOUT,self.OnAbout)
        EVT_MENU(self,ID_EXIT,self.OnExit)
        #membuat menubar dan menempatkan
        menufile pada menubar
        menubar = wxMenuBar()
        menubar.Append(menufile, "&File")
        #mengaktifkan menubar
        self.SetMenuBar(menubar)
        def OnAbout(self,event):
            dlg = wxMessageDialog(self, "Maaf,
```



▲ Gambar 3. Menu 1, 2, 3.

```
informasi program belum tersedia",
"About",wxOK|wxICON_INFORMATION)
dlg.ShowModal()
dlg.Destroy()
def OnExit(self,event):
dlg = wxMessageDialog(self, "Apa Anda
yakin untuk keluar ?" ,"Exit",wxYES_NO|
wxICON_INFORMATION)
if dlg.ShowModal() == wxID_YES:
self.Close(1)
dlg.Destroy()

app = wxPySimpleApp()
frame = main_window(None,-1, "Hello
World")
frame.Show(1)
app.MainLoop()
```

Pada saat program dijalankan, kita mengimpor modul wx dari wxPython, kemudian kita membuat ID untuk menu *About* dan *Exit*. ID berfungsi sebagai pengenalan pada saat memproses event pada menu. Selanjutnya kita membuat kelas *main_window* yang diturunkan dari kelas *wxFrame* dan membuat statusbar serta menambahkan item ke dalam menufile. Selanjutnya kita akan menempatkan menufile pada menu bar, kemudian menu bar kita aktifkan.

Di dalam kelas *main_window* kita juga membuat fungsi *OnAbout()* dan *OnExit()*. Apabila kita mengklik menu *About*, maka akan ditampilkan message dialog yang berisi informasi tertentu. Dan apabila kita mengklik menu *Exit*, program akan menampilkan *message dialog* untuk mengkonfirmasi apakah Anda yakin untuk keluar dari program atau tidak. Baris selanjutnya seperti biasanya kita menginstansiasi objek app dengan kelas *wxPySimpleApp*, kemudian membuat objek frame yang diturunkan dari

kelas *main_window* dan mengaktifkannya. Kemudian kita memanggil *MainLoop* untuk melakukan proses *looping* sampai program kita ditutup. Hasil program akan tampak seperti Gambar 3.

wxPanel()

Panel berupa sebuah window yang berisi kumpulan kontrol-kontrol tertentu. Biasanya panel ditempatkan di dalam frame.

wxButton()

Unit kontrol ini berupa tombol untuk menjalankan fungsi tertentu. Untuk membuat sebuah button di program Anda, hal tersebut dapat dengan mudah dibuat.

Event handling:

EVT_BUTTON(kontrol,ID,fungsi)

→ Event ini dipanggil ketika user mengklik tombol tertentu.

```
...
tombol = wxButton(self,20, "OK",
wxPoint(20,30))
tombol.SetDefault()
tombol.SetToolTipString("Ini adalah
tombol OK")
# Event ini dipanggil ketika tombol diklik.
EVT_BUTTON(self,20,self.Onclick)
...
```

Pada potongan kode di atas, kita membuat suatu objek tombol (*button*) dengan *caption* "OK" ID 20 pada posisi(20,30) dalam window. Kemudian apabila kita mengklik tombol OK tersebut akan dijalankan fungsi *OnClick()* yang kita buat. Baris selanjutnya kita membuat tombol OK sebagai tombol *default* pada program kita. Baris terakhir kita menambahkan tips pada tombol yang akan ditampilkan ketika pointer mouse berada di atas tombol tersebut.

wxCheckBox()

Unit kontrol ini digunakan untuk memasukkan data pilihan yang berupa *True/False*.

Event Handling:

EVT_CHECKBOX(kontrol,ID,fungsi)

→ Event ini dipanggil ketika user memilih/mengklik checkbox tersebut.

wxComboBox()

Unit kontrol yang merupakan kombinasi dari text control dan listbox, di mana user dapat mengisinya dengan mengetik pilihan ataupun memilih item dari *drop-down list*.

Event Handling:

EVT_COMBOBOX(kontrol,ID,fungsi)

→ Event ini dipanggil ketika user memilih item dari combo box.

EVT_TEXT(kontrol,ID,fungsi)

→ Event ini dipanggil ketika user mengetikkan pilihannya dalam combo box atau user memilih salah satu item dari combo box.

wxRadioBox()

Berupa sekelompok option *radio button* di mana user hanya boleh memilih satu item dari semua pilihan.

Event Handling :

EVT_RADIOBOX(kontrol,ID,fungsi)

→ Event ini dipanggil ketika user memilih salah satu item dalam radiobox tersebut.

wxStaticText()

Unit kontrol ini berfungsi untuk menampilkan teks tertentu di mana user tidak dapat mengubahnya.

wxTextCtrl()

Unit kontrol ini berguna untuk menampilkan

teks di mana user dapat mengubah isinya, akan tetapi bisa diset propertinya menjadi *read-only* agar tidak bisa diedit user.

Event Handling:

EVT_TEXT(kontrol,id,fungsi)

→ Event ini dipanggil ketika teksnya berubah.

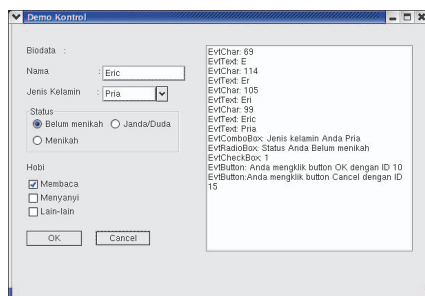
EVT_CHAR(kontrol,fungsi)

→ Event ini dipanggil untuk menandakan bahwa suatu tombol ditekan.

Berikut kita akan membuat program yang menggunakan kontrol-kontrol di atas beserta contoh penanganan event-nya.

```
from wxPython.wx import *

class Form(wxPanel):
    def __init__(self, parent, id):
        wxPanel.__init__(self, parent, id)
        # Membuat text control yang multi
        # line dan read-only(tidak bisa diedit)
        self.log = wxTextCtrl(self, 5, "",
                               wxPoint(280,30), wxSize(300,300),
                               wxTE_MULTILINE|wxTE_READONLY)
        self.biadata = wxStaticText(self,
                                     1,"Biadata : ",wxPoint(20, 30))
        # Button
        self.button1 = wxButton(self, 10,
                                "OK", wxPoint(20,300))
        self.button2 = wxButton(self, 15,
                                "Cancel",wxPoint(120,300))
        EVT_BUTTON(self, 10, self.
                   OnClickOK)
        EVT_BUTTON(self,15,self.OnClick
                   Cancel)
        # Static Text & Text Control
        self.lblname = wxStaticText(self,
                                     2,"Nama : ",wxPoint(20,60))
        self.editname = wxTextCtrl(self,
                                    20,"", wxPoint(130, 60),
                                    wxSize(120,-1))
        EVT_TEXT(self, 20, self.EvtText)
```



Gambar 4. Demo kontrol.

```
EVT_CHAR(self.editname, self.
EvtChar)
# ComboBox
self.listgender = ['Pria', 'Wanita']
self.lblgender = wxStaticText
(self,3,"Jenis Kelamin :",
wxPoint(20, 90))
self.gender = wxComboBox(self, 30,
"", wxPoint(130, 90), wxSize
(95, -1),
self.listgender, wxCB_DROPDOWN)
EVT_COMBOBOX(self, 30,
self.EvtComboBox)
EVT_TEXT(self, 30, self.EvtText)
# Checkbox
self.hobi = wxStaticText(self,3,
"Hobi",wxPoint(20,200))
self.baca = wxCheckBox(self,
40,"Membaca",wxPoint(20,220))
self.nyanyi = wxCheckBox(self,50,
"Menyanyi",wxPoint(20,240))
self.lain = wxCheckBox(self,60,
"Lain-lain",wxPoint(20,260))
EVT_CHECKBOX(self, 40,self.
EvtCheckBox)
EVT_CHECKBOX(self, 50,self.
EvtCheckBox)
EVT_CHECKBOX(self, 60,self.
EvtCheckBox)
# RadioBox
self.radioList = ['Belum menikah',
'Menikah','Janda/Duda']
rb = wxRadioBox(self, 70, "Status",
wxPoint(20, 120), wxDefaultSize,
self.radioList, 2, wxRA_
SPECIFY_COLS)
EVT_RADIOBOX(self, 70,
self.EvtRadioBox)

def EvtRadioBox(self, event):
    self.log.AppendText('EvtRadioBox:
Status Anda %s\n' % event.
GetString())

def EvtComboBox(self, event):
    self.log.AppendText('EvtComboBox:
Jenis kelamin Anda %s\n' %
event.GetString())

def OnClickOK(self,event):
    self.log.AppendText('EvtButton:
Anda mengklik button OK dengan ID
%s\n' % event.GetId() )

def OnClickCancel(self,event):
    self.log.AppendText('EvtButton:
Anda mengklik button Cancel
```

```
dengan ID %s\n' % event.GetId() )
def EvtText(self, event):
    self.log.AppendText('EvtText: %s\n'
% event.GetString())
def EvtChar(self, event):
    self.log.AppendText('EvtChar:
%d\n' % event.GetKeyCode())
    event.Skip()
def EvtCheckBox(self, event):
    self.log.AppendText('EvtCheckBox:
%d\n' % event.Checked())

app = wxPySimpleApp()
frame = wxFrame(None, -1, "Demo
Kontrol",wxDefaultPosition,wxSize(600,400))
Form(frame,2)
frame.Show(1)
app.MainLoop()
```

Hasil program di atas akan tampak seperti gambar 4.

Pertama-tama, kita membuat kelas `Form` yang merupakan turunan dari kelas `wxPanel`, selanjutnya kita menambahkan kontrol-kontrol button, combo box, radio box, check box ke dalam panel `Form`. Kita juga membuat objek log yang berupa text control untuk menampilkan event-event yang berhubungan dengan kontrol-kontrol yang ada. Apabila kita mengakses kontrol tertentu maka akan ditampilkan event-event-nya di dalam objek log. Untuk memproses event kontrol tertentu, kita menggunakan ID pengenalan dari setiap kontrol yang kita buat.

Berikutnya kita menginstansiasi objek `app` dengan kelas `wxPySimpleApp`, kemudian membuat objek `frame` yang diturunkan dari kelas `wxFrame` dan menempatkan objek `Form` ke dalam `frame`. Kemudian kita memanggil `MainLoop` untuk melakukan proses *looping* sampai program kita ditutup.

Setelah Anda melihat contoh-contoh program di atas, ternyata membuat aplikasi GUI dengan wxPython tidaklah sesulit yang Anda bayangkan, bukan? wxPython menawarkan berbagai kemudahan dan kelebihan sehingga bukan lagi merupakan hambatan bagi pemula untuk mempelajarinya. Kini pemrograman GUI yang multi-platform telah didukung oleh wxPython, apa lagi yang Anda tunggu untuk segera beralih ke wxPython? [Eric \(eric85id@plasa.com\)](mailto:eric85id@plasa.com)