Palm OS Programming - C Tips

INDEX

Gotchyas
Advanced UI
PilRC
C Code
Events

To save space, you should go against all of the examples for your form's event loop and return true/false as soon as you know it. It also speeds up your program a *very* minor amount. Lastly, 'if' statements are compiled to a smaller amount of code than a switch/case method.

```
// Bad example
Boolean BadFormEventHandler(EventPtr event)
{
    Boolean handled = false;
    LocalID id;

    switch (event->eType)
    {
        case frmOpenEvent:
            // ... insert code here ...
            handled = true;
            break;

        case ctlSelectEvent:
            id = event->data.ctlEnter.controlID;

            if (id == Button1_ID)
            {
                // ... insert code here ...
                handled = true;
            }
            else if (id == Button2_ID)
            {
                // ... insert code here ...
                handled = true;
            }
    }

    return handled;
}


// Good example
Boolean GoodFormEventHandler(EventPtr event)
{
    LocalID id;

    if (event->eType == frmOpenEvent)
    {
        // ... insert code here ...
        return true;
    }
    if (event->eType == ctlSelectEvent)
    {
        id = event->data.ctlEnter.controlID;
        if (id == Button1_ID)
        {
            // ... insert code here ...
            return true;
        }
```

```
        if (id == Button2_ID)
        {
            // ... insert code here ...
            return true;
        }
    }
    return false;
}
```

Some people will put an 'else' right after a 'return' statement. This is silly. Not only does it create a larger program, but it could cause errors to spring up. As a general rule of thumb, if you return from an 'if' block, don't use an 'else' block.

```
// Bad example
if (some_thing == what_you_want)
{
    // ... insert code here ...
    return true;
}
else   // 'else' is not needed.
{
    // ... insert code here ...
    return false;
}
// Any code here will never get executed.


// Good example
if (some_thing == what_you_want)
{
    // ... insert code here ...
    return true;
}
// Maybe add a comment here so you know that
// this is the 'else' portion
// ... insert code here ...
return false;
```

Along the same lines, if you return from the 'else' and not the 'if' section, rewrite it so that the 'else' part is first, then return from the 'if' section.

```
// Bad
if (A < 0.5)
{
    // ... insert < 0.5 code here ...
}
else
{
    // ... insert >= 0.5 code here ...
    return false;
}
// ... more code here ...
return true;


// Good
if (A >= 0.5)
{
    // ... insert >= 0.5 code here ...
    return false;
```

```
}
// ... insert < 0.5 code here ...
// ... more code here ...
return true;
```

When initializing large arrays or structs, you should declare them 'static' if they never change. If you don't, they will be created as dynamic variables, and will possibly not be initialized correctly. This appears to happen when you hit about 64 bytes. Therefore, if you have any lookup tables that you just use to reference information, declare them static.

```
// Good example of my form loading function
typedef struct {
    LocalID formID;
    FormEventHandlerType *handler;
} FormHandlerStruct;

void FormLoadEvent(EventPtr event)
{
    LocalID formID;
    FormPtr form;
    int i;
    static FormHandlerStruct FormToHandler[] =
    {
        { F_StartScreen, StartScreenEventHandler },
        { F_Preferences, PreferencesEventHandler },
        { F_AnotherForm, AnotherFormEventHandler },
        { F_About, AboutFormEventHandler },
        { 0, NULL }
    };

    formID = event->data.frmLoad.formID;
    form = FrmInitForm(formID);
    FrmSetActiveForm(form);

    for (i = 0; FormToHandler[i].formID != 0; i ++)
    {
        if (FormToHandler[i].formID == formID)
        {
            FrmSetEventHandler(form, FormToHandler[i].handler);
            return;
        }
    }
    // If you go this far, you forgot to associate the form handler
    // with the form ID.
}
```


Tyler Akins <fidian@rumkin.com>