Advanced Igor Programming

Leon Camenzind

FAM Tech Talk
7.11.2014
Movivation

• Igor Pro used on a daily basis
• Give you some new ideas/possibilities
• Show you Igor Pro’s potential
Outline

• Igor Pro:
  – Useful operations
  – Panels
  – eXternal Operation (XOP) Introduction
  – Background-Tasks

• Project Management Software: Wrike
USEFUL OPERATIONS IN IGOR YOU MAY NOT BE AWARE OF
Curve Fit / FuncFit

- CurveFit [ flags ] fitType, [kwCWave=coefWaveName ,] waveName

- «Performance» flags:

  - \N=1 Suppress all Igor updates during fit!

  **Curve Fit Results Box**
  - \W=Wait
    - W=0 Show but not wait (useless but default)
    - W=1 Wait till users clicks ok
    - W=2 **Do not display curve results at all!**

Huge gain (up to 50x) when used inside a loop!
Optional Parameters

Use optional parameters!

Call function with optional parameters:
• You can still call `testFitWithOrWithout()`
• `print testFitWithOrWithout(W=2,N=1)`
• All Data types (string, variable, Wave, DFREF) possible
• Also useful for functions with plenty of parameters (because Parameter=...)

MSTimer: \( \mu s \) resolution
Ticks: 1/60s resolution

```plaintext
function testFitWithWaiting()
    variable i
    variable t1 = StartMsTimer
    Wave w_coef, mycurve
do
    FuncFit/Q fermi W_coef, mycurve /D
    i+=1
while(i<1000)
return StopMsTimer(t1)
end

function testFitWithoutWaiting()
    variable i
    variable t1 = StartMsTimer
    Wave w_coef, mycurve
do
    FuncFit/Q/W=2/N=1 fermi W_coef, mycurve /D
    i+=1
while(i<1000)
return StopMsTimer(t1)
end
```
Concept of State Machines

Based on output of processing it is decided if you use CaseA, CaseB or Stop.
A Simple Igor State Machine

function StateMachine()

variable i
string decision = "CaseA"
do
strswitch(decision)
case "CaseA":
  print "got CaseA"
  break
case "CaseB":
  print "got CaseB"
  break
default:
  print "got neither A nor B"
  break
endswitch
DoDecision(decision)
i+=1
while(i<10)
end

Initialization value

decision = "CaseA" or "CaseB"

Reference!

function DoDecision(decision)

string &decision

variable random = abs(enoise(1))

if(random<0.5)
  decision = "CaseA"
else
  decision = "CaseB"
endif

Creates a random number (0,1)
→ Decision how to proceed
ORGANISATION WITH PANELS
Accessing a graph inside a panel:

- ModifyGraph/W=Panel0#G0 rgb=(0,15872,65280)
- You can save Panel as Macro for recreation
- Graphs in Panels are like «normal» graphs
Cockpit organisation
Active Panel elements

```
Hello world!
```
Example

DoSensor()
Introduction to

EXTERNAL OPERATIONS (XOP)
• XOP acts like a «Pipe» to outside world
• Makes Igor Pro extremely powerful
• Needs **decent programming skills** in C/C++ and Igor Pro
• For certain interfaces there are written packages
If there is no standard protocol you can still use XOPs to access the driver
• Basically every device is shipped with a driver
• Potential optimizations for the device/operation used
BACKGROUND TASKS

Loop 1
(Aquiring data)

Loop 2
(Analyse data in background)

Main loop

?
Set up and run

Function TestTask(s)
    // This is the function that will be called periodically
    STRUCT WMBackgroundStruct &s
    Printf "%s called, Time=%s\n", s.name, Time()
    return 0 // Continue background task
End

Function StartTestTask()
    Variable numTicks = 2 * 60 // Run every two seconds (120 ticks)
    CtrlNamedBackground Test, period=numTicks, proc=TestTask
End

Function StopTestTask()
    CtrlNamedBackground Test, stop
End

Similar to C++ «classes»
Initialize Background task
Background tasks

- **Limitation:**
  Is not **executed** when Igor is inside a loop
- Useful for periodic operations when not measuring
- You can run multiple (named) background tasks
- Background task for data acquisition using XOP

Collect data constantly in background
XOPIdle → Igor gets data from driver (XOP)
XOP runs independently of Igor main loop

Costs 20ms to create task
Parallel Loops

Loop 1 (e.g. Acquiring data)

Loop 2 (e.g. Analyse data in background)

Not even possible using background tasks or/and independent modules

Loop 1

Loop 2 In XOP

You can even push data (e.g. Waves) from Loop 2 to Loop 1
Conclusion

• Some new ideas/concepts
• Learning by doing
• Igor Pro has become even more powerful by supporting XOPs

• Check out http://www.igorexchange.com/
Concept of Master / Slave structure

This template is for the Producer/Consumer design pattern.

This loop is the producer loop.

Compute conditions to decide whether data is required.

Generate the data here.

Releasing the queue stops the consumer loop(s).

This loop is a consumer loop.

No Error

'element'