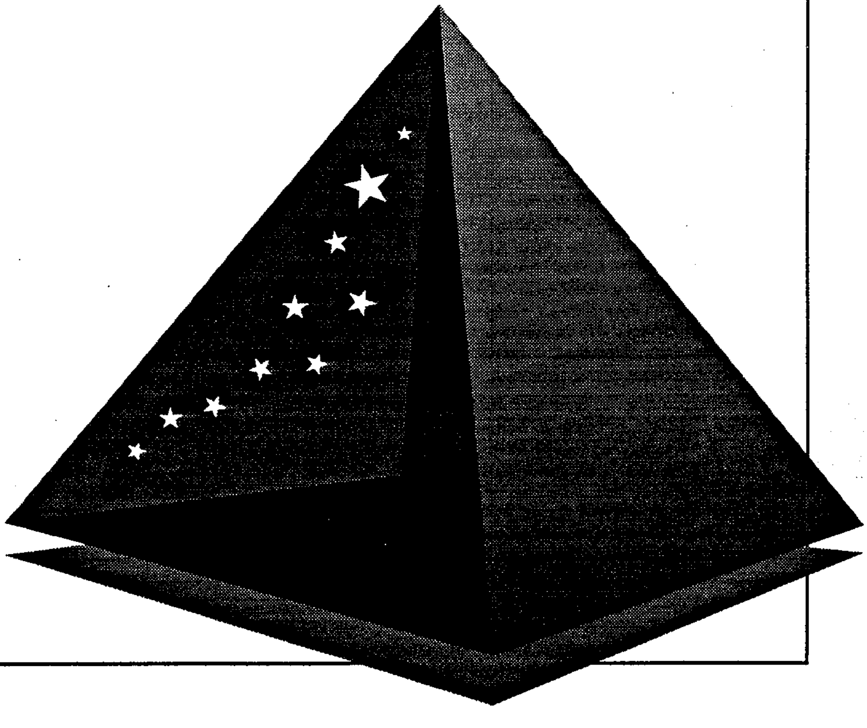
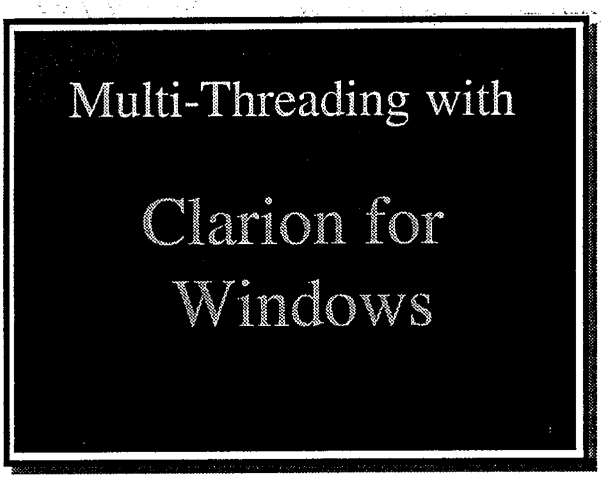
DEVCON '94

MULTI-THREADING IN CLARION FOR WINDOWS

Ross Santos

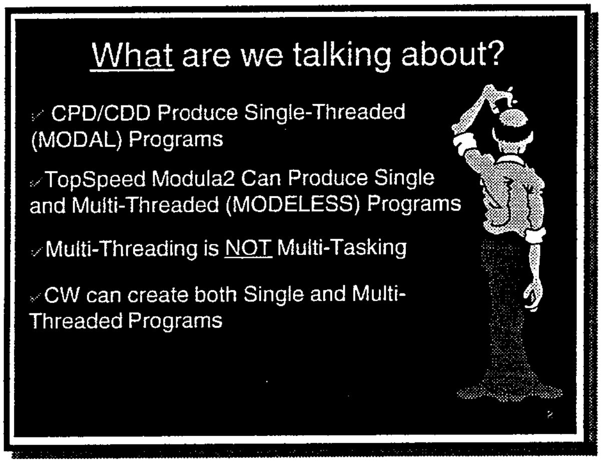


TopSpeed™
CORPORATION



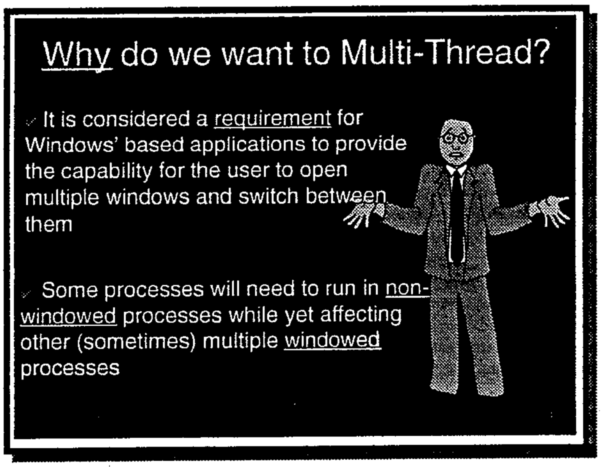
Multi-Threading with

Clarion for Windows



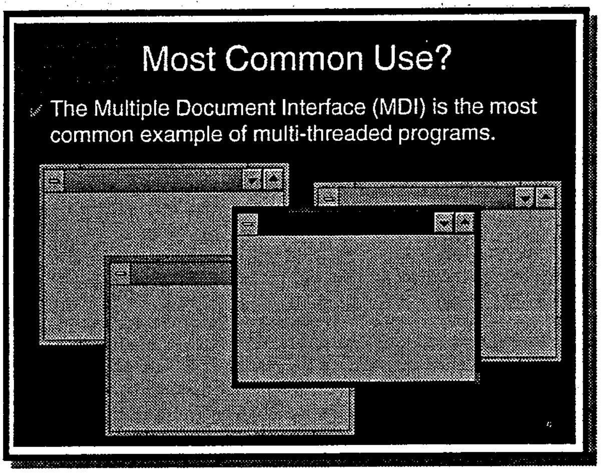
What are we talking about?

- ✓ CPD/CDD Produce Single-Threaded (MODAL) Programs
- ✓ TopSpeed Modula2 Can Produce Single and Multi-Threaded (MODELESS) Programs
- ✓ Multi-Threading is NOT Multi-Tasking
- ✓ CW can create both Single and Multi-Threaded Programs



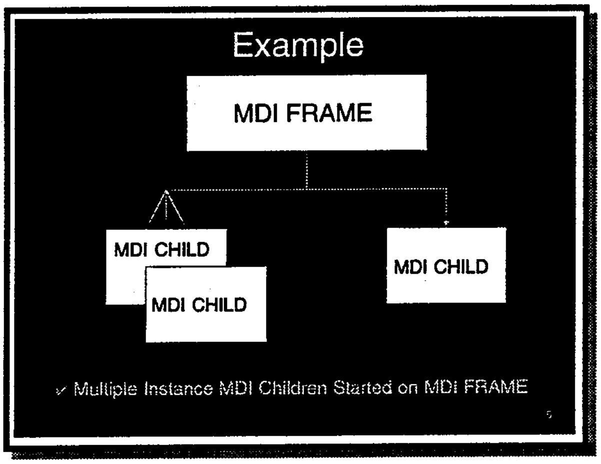
Why do we want to Multi-Thread?

- ✓ It is considered a requirement for Windows' based applications to provide the capability for the user to open multiple windows and switch between them
- ✓ Some processes will need to run in non-windowed processes while yet affecting other (sometimes) multiple windowed processes



Most Common Use?

- ✓ The Multiple Document Interface (MDI) is the most common example of multi-threaded programs.



Example

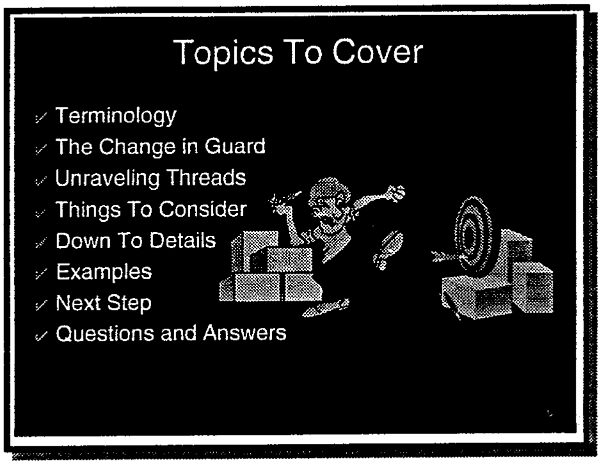
MDI FRAME

MDI CHILD

MDI CHILD

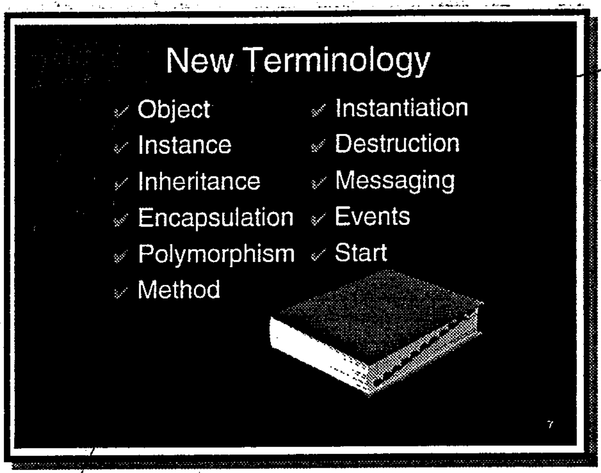
MDI CHILD

- ✓ Multiple Instance MDI Children Started on MDI FRAME



Topics To Cover

- ✓ Terminology
- ✓ The Change in Guard
- ✓ Unraveling Threads
- ✓ Things To Consider
- ✓ Down To Details
- ✓ Examples
- ✓ Next Step
- ✓ Questions and Answers



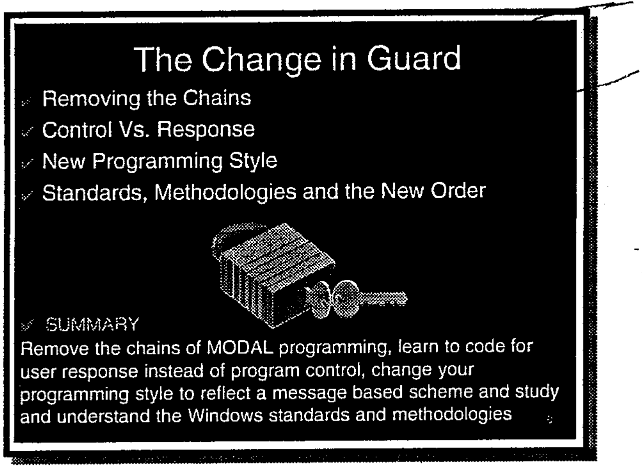
New Terminology

- ✓ Object
- ✓ Instance
- ✓ Inheritance
- ✓ Encapsulation
- ✓ Polymorphism
- ✓ Method
- ✓ Instantiation
- ✓ Destruction
- ✓ Messaging
- ✓ Events
- ✓ Start

OBJect – Data/Process or Both

To start a Fread

Func/Pro



The Change in Guard

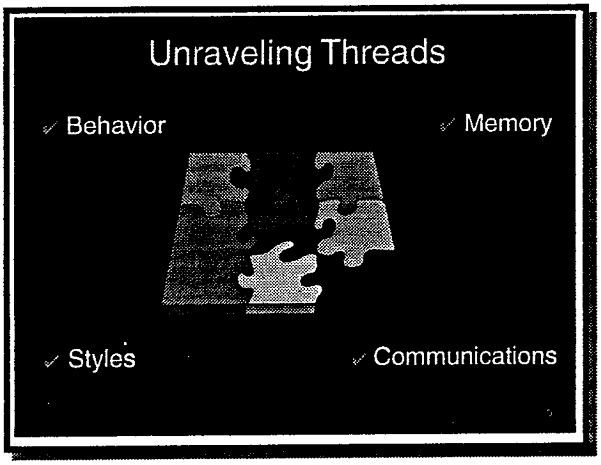
- ✓ Removing the Chains
- ✓ Control Vs. Response
- ✓ New Programming Style
- ✓ Standards, Methodologies and the New Order

✓ SUMMARY

Remove the chains of MODAL programming, learn to code for user response instead of program control, change your programming style to reflect a message based scheme and study and understand the Windows standards and methodologies

old logic is linear

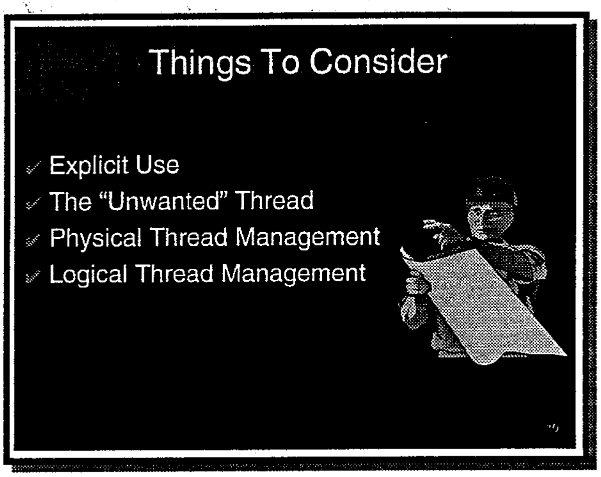
in Win the User is in Control and you Respon



Unraveling Threads

- ✓ Behavior
- ✓ Memory
- ✓ Styles
- ✓ Communications

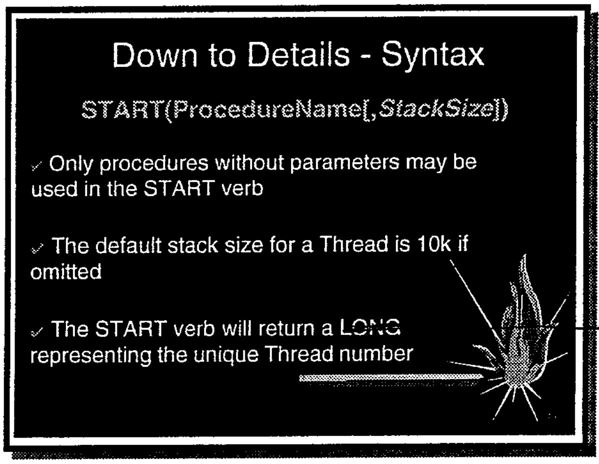
style



Things To Consider

- Explicit Use
- The "Unwanted" Thread
- Physical Thread Management
- Logical Thread Management

Limited to 64 threads

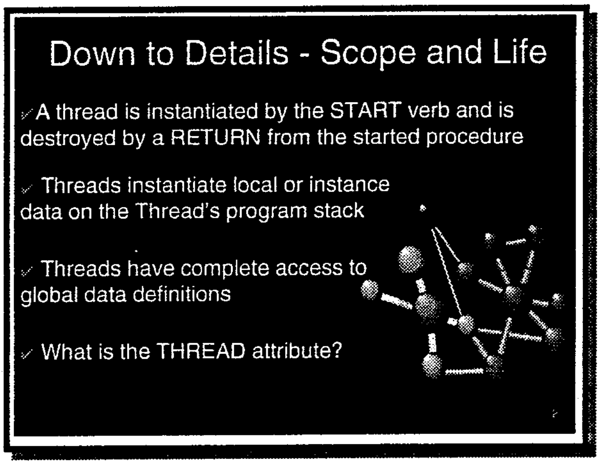


Down to Details - Syntax

START(ProcedureName[,*StackSize*])

- Only procedures without parameters may be used in the START verb
- The default stack size for a Thread is 10k if omitted
- The START verb will return a LONG representing the unique Thread number

USHORT

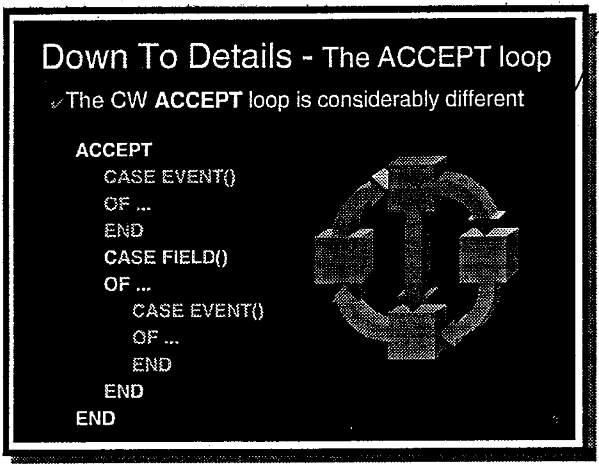


Down to Details - Scope and Life

- A thread is instantiated by the START verb and is destroyed by a RETURN from the started procedure
- Threads instantiate local or instance data on the Thread's program stack
- Threads have complete access to global data definitions
- What is the THREAD attribute?

THREAD - New instances of Data per thread. Can Assign to Record Buffer

VARIA STRING, THREAD

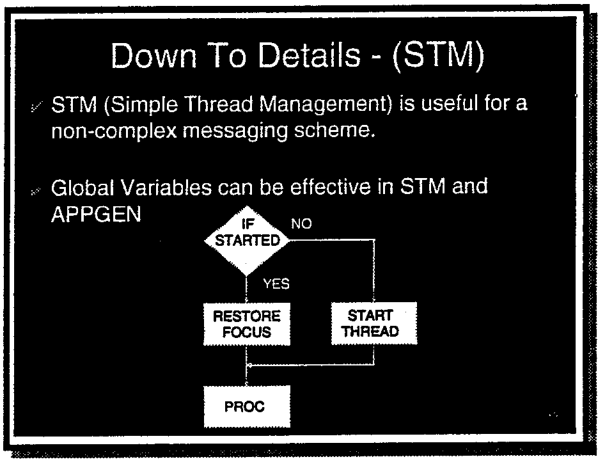


Down To Details - The ACCEPT loop

- ✓ The CW **ACCEPT** loop is considerably different

```
ACCEPT
   CASE EVENT()
   OF ...
   END
   CASE FIELD()
   OF ...
      CASE EVENT()
      OF ...
      END
   END
END
```

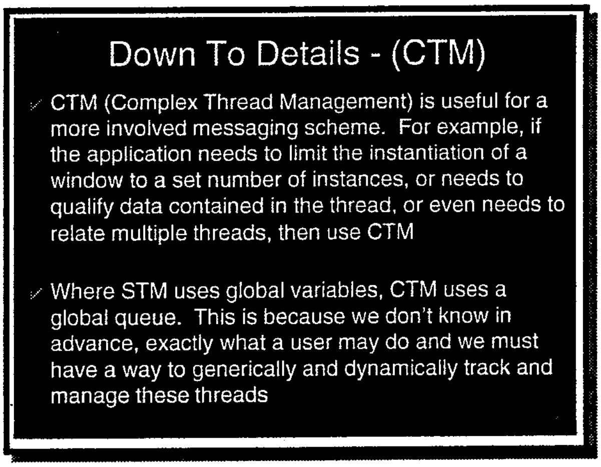
Global Events



Down To Details - (STM)

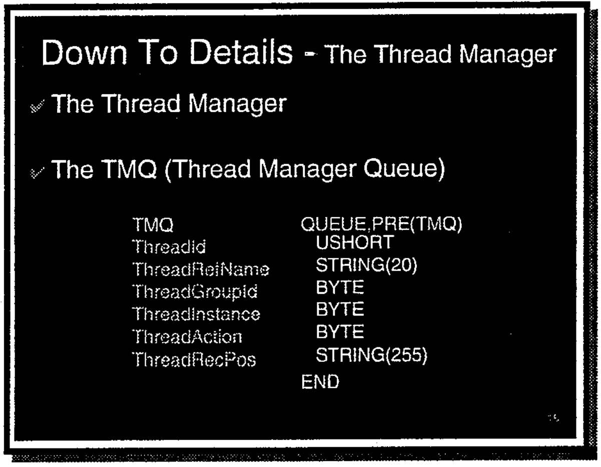
- ✓ STM (Simple Thread Management) is useful for a non-complex messaging scheme.
- ✓ Global Variables can be effective in STM and APPGEN

IF STARTED
NO
YES
RESTORE FOCUS
START THREAD
PROC



Down To Details - (CTM)

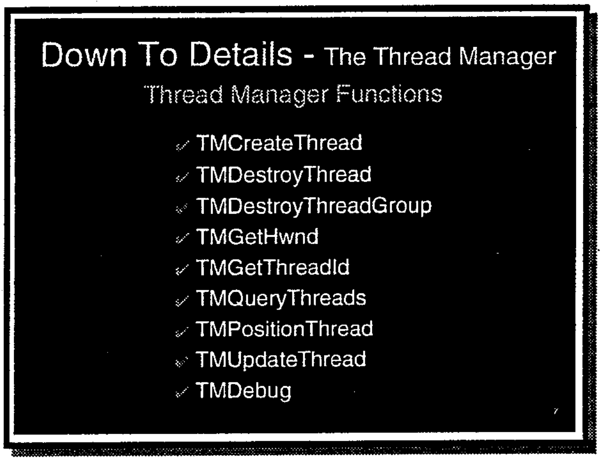
- ✓ CTM (Complex Thread Management) is useful for a more involved messaging scheme. For example, if the application needs to limit the instantiation of a window to a set number of instances, or needs to qualify data contained in the thread, or even needs to relate multiple threads, then use CTM
- ✓ Where STM uses global variables, CTM uses a global queue. This is because we don't know in advance, exactly what a user may do and we must have a way to generically and dynamically track and manage these threads



Down To Details - The Thread Manager

- ✓ The Thread Manager
- ✓ The TMQ (Thread Manager Queue)

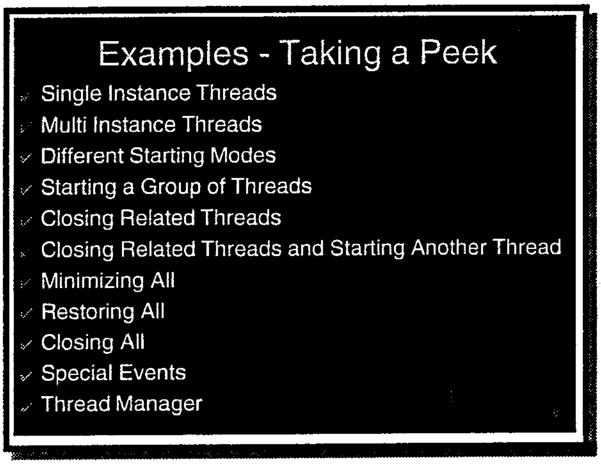
```
TMQ                QUEUE,PRE(TMQ)
Threadid             USHORT
ThreadRefName        STRING(20)
ThreadGroupid        BYTE
ThreadInstance       BYTE
ThreadAction         BYTE
ThreadRecPos         STRING(255)
                   END
```



Down To Details - The Thread Manager

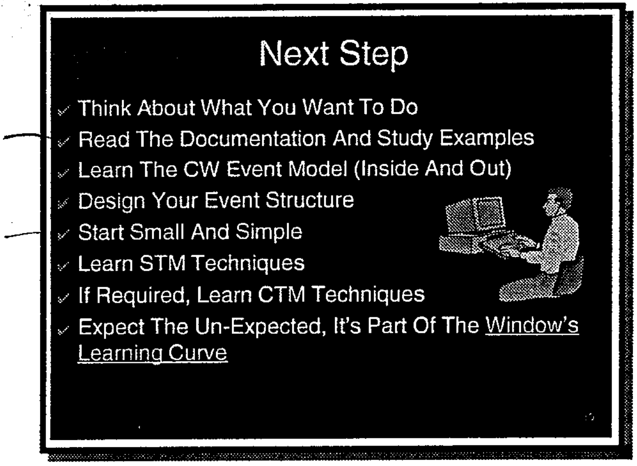
Thread Manager Functions

- ✓ TMCreateThread
- ✓ TMDestroyThread
- ✓ TMDestroyThreadGroup
- ✓ TMGetHwnd
- ✓ TMGetThreadId
- ✓ TMQueryThreads
- ✓ TMPositionThread
- ✓ TMUpdateThread
- ✓ TMDebug



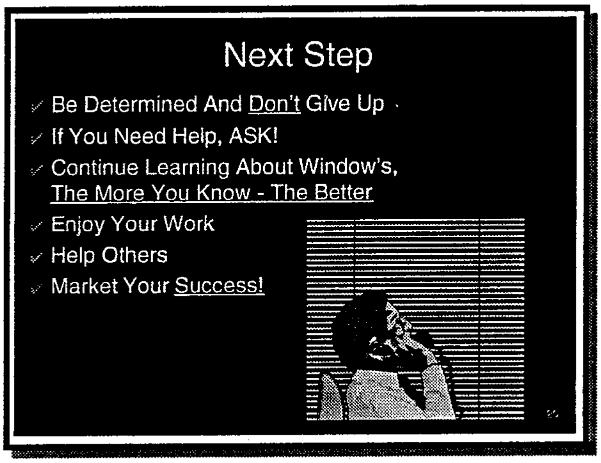
Examples - Taking a Peek

- ✓ Single Instance Threads
- ✓ Multi Instance Threads
- ✓ Different Starting Modes
- ✓ Starting a Group of Threads
- ✓ Closing Related Threads
- ✓ Closing Related Threads and Starting Another Thread
- ✓ Minimizing All
- ✓ Restoring All
- ✓ Closing All
- ✓ Special Events
- ✓ Thread Manager



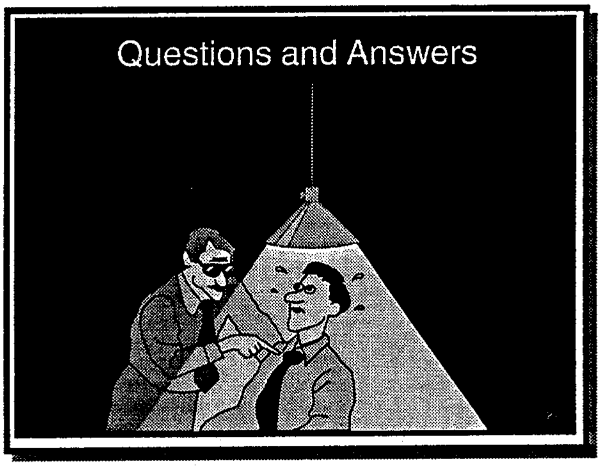
Next Step

- ✓ Think About What You Want To Do
- * ✓ Read The Documentation And Study Examples
- ✓ Learn The CW Event Model (Inside And Out)
- ✓ Design Your Event Structure
- * ✓ Start Small And Simple
- ✓ Learn STM Techniques
- ✓ If Required, Learn CTM Techniques
- ✓ Expect The Un-Expected, It's Part Of The Window's Learning Curve

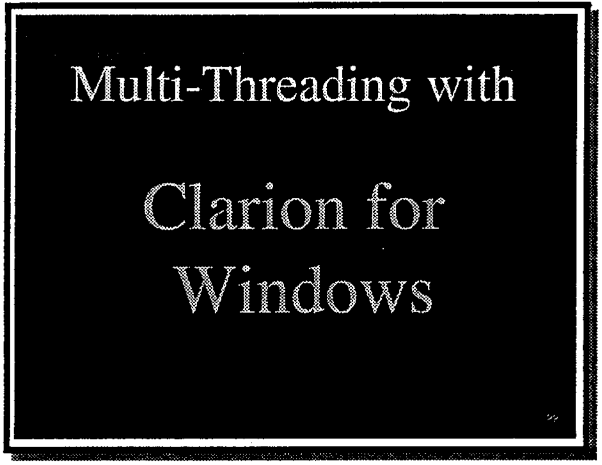


Next Step

- ✓ Be Determined And Don't Give Up
- ✓ If You Need Help, ASK!
- ✓ Continue Learning About Window's, The More You Know - The Better
- ✓ Enjoy Your Work
- ✓ Help Others
- ✓ Market Your Success!



Questions and Answers



Multi-Threading with

Clarion for Windows