Application Note

Imagenation **PXC200A**

Multi-threaded Display Applications

Written in C & C++

Overview This application note discusses multi-threaded software development as it relates to the Imagenation PXC200A frame grabber and gives two useful examples that you may use as the basis for further development. Multi-threaded software has several advantages and disadvantages when compared to single-threaded software. It should not be used in every application, but there are times when it makes sense.

Advantages

1. Lets a complex application be divided into logically separate tasks to simplify design
2. Individual threads of execution seem to run concurrently
3. Lengthy tasks can run in the background

Disadvantages

1. Programs can be difficult to debug if not designed properly
2. Threads can result in race conditions without proper synchronization
3. Interaction between threads may have unpredictable results without proper controls

The two programs presented here have exactly the same functionality: continuous display. One is written in C and the other in C++. A multi-threaded program written in C has the disadvantage of being more time-consuming to write but the advantage of being more open with details. On the other hand, a multi-threaded program written in C++ using MFC (Microsoft Foundation Classes) is probably easier to write but the threading details are hidden. Neither language is better than the other; the choice is simply a matter of taste.

The best way to use this application note is to decide which language you like and then study that source code. There is probably no point in comparing the two.

Program description All programs have at least one thread of execution, referred to as the main thread. In addition to the main thread, the two programs under discussion here are using a second thread to control the display of images. The second, the display thread, is completely controlled by the main thread. It is started, stopped, and paused by the main thread in response to user actions.

There are two types of events that can be used in multi-threaded programs: manual and automatic. Both of the programs discussed in this note use manual events, because they are easier to understand and control. Automatic events perform automatic resets and can present synchronization problems.

Imagenation PXC200A

C version The source code for the display thread is the last function in the source file and is named “DisplayThreadProc.” It is a very simple forever loop, written “for(;;),” with three blocks of code. The first block waits for a “pause,” the second block waits for an “exit,” and the third blocks displays an image. The third block of code was lifted from our standard sample PXCDRAW2 and won’t be discussed in this note.

The first block of code that waits for a “pause” event has a wait time of zero, so it checks for a “pause” event and continues if one has not been issued by the main thread. If the “pause” event was issued, the display thread then waits forever for a “continue” event. After the “continue” event is received, both events: “pause” and “continue” are reset, so they can’t be accidentally set twice by a bouncing mouse or keyboard.

The second block of code has a wait time of zero and simply checks for the “exit” event. If the exit event has not been set, the display thread displays another image. If the “exit” event has been set, then it is reset and the forever loop is broken which ends the function and terminates the thread.

The important points to study in this program are:

Thread control

Thread creation under “case WM_CREATE”
Thread termination under “case WM_DESTROY”

Event control

Event creation in function AppIint()
Event destruction by the closing of handles in function AppExit()

Event processing

WaitForSingleObject under “case ID_ACQUIRE”
WaitForSingleObject under “case ID_STOP”
WaitForSingleObject under “case ID_WRITE”
WaitForSingleObject under “case ID_READ”

C++ version The source code for the display thread is the last function in the source file CPXC.CPP and is named “DisplayThreadProc.” It is a very simple forever loop, written “for(;;),” with three blocks of code. The first block waits for a “pause,” the second block waits for an “exit,” and the third blocks displays an image. The third block of code won’t be discussed in this note.

The first block of code that waits for a “pause” event has a wait time of zero, so it checks for a “pause” event and continues if one has not been issued by the main thread. If the “pause” event was issued, the display thread then waits forever for a “continue” event. After the “continue” event is received, both events: “pause” and “continue” are reset, so they can’t be accidentally set twice by a bouncing mouse or keyboard.

The second block of code has a wait time of zero and simply checks for the “exit” event. If the exit event has not been set, the display thread displays another image. If the “exit” event has been set, then it is reset and the forever loop is broken which ends the function and terminates the thread.

The important points to study in this program are:

Thread control

Thread creation in class function “OnInitialUpdate()” in file CPXC_GRAB_MFCView.cpp
Thread termination in class function “AppExit()” in file CPXC.cpp

Event control

Event creation at the top of the file CPXC_GRAB_MFCView.cpp
Event destruction is handled by the event class and is invisible.

Event processing

WaitForSingleObject in class function ACQUIRE()
WaitForSingleObject in class function STOP()
WaitForSingleObject in class function OnFileOpen()
WaitForSingleObject in class function OnFileSave()
All of the above class functions are in the file CPXC_GRAB_MFCView.cpp

Conclusion If you think multi-threading is appropriate for your application, each of these two programs can be modified to do something more interesting than just display by the addition of threads. For example, you might add a thread to perform image processing.

Related samples

The PXCDRAW2 sample can be found at the following Web address:

http://www.imagenation.com/dnpages/pxc_files.html
Heading:PXC Example Source Code (Win32 and C/C++)
Title: PXCDRAW

Telephone Service

Toll Free:	800-366-9131
Phone:	503-495-2200
Fax:	503-495-2201

Online Service

Email:	CSsupport@cyberoptics.com
Web:	http://www.CyberopticsSemi.com

CyberOptics Semiconductor
13555 SW Millikan Way
Beaverton, OR 97005