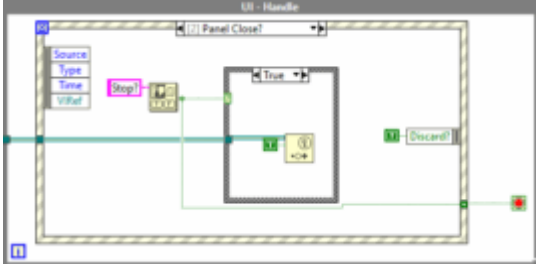


Evitare il polling nelle interfacce utente

Category: LabVIEW Talk, Portfolio
scritto da admin | Settembre 11, 2021



Polling

La tecnica del polling (sondaggio), consiste nel verificare continuamente lo stato di un ingresso(es. controllo sull'interfaccia grafica, a cadenza definita da un loop temporizzato (WAIT(ms) e WAIT UNTIL NEXT ms MULTIPLE)).

Questa tecnica la troviamo spesso su codice vecchio o non professionale.

Una programmazione professionale, tende ad eliminare quanto più possibile, l'utilizzo del polling.

Scheduler

Lo scheduler è gestito a livello applicativo e di sistema operativo. Se si pensa al multitasking, un'applicazione richiede attività solo quando c'è un'azione richiesta o allo scatenarsi di un intervallo di tempo ozioso (Idle), chiamato timeout.

Utilizzando un dialogo con lo scheduler, diretto, non si avrebbe più necessità di controllare a tempi regolari lo stato degli input dell'interfaccia grafica, risparmiando utilizzi e richieste di CPU inutili, in quanto siamo in idle, in attesa che si verifichi un evento. Eventualmente, potremmo definire un timeout molto lungo per eseguire qualcosa in caso si prolunghi l'attesa in idle.

Uno scenario nella vita comune, l'impiegata delle poste, serve schedulando per ordine di arrivo le richieste dei clienti in coda. Esaurita la coda, l'impiegata entra in Idle, aspetta, dopo 30minuti che non arriva nessuno (timeout) decide di prendersi una piccola pausa per un caffè, poi si rimette in attesa, o se nel frattempo sono arrivati clienti, inizia a schedulerli.

Immaginate se l'impiegata invece fosse in polling? Niente caffè perché il tempo di polling non glielo permette e deve chiedere "avanti il prossimo" anche quando non c'è nessuno in fila.

Event Driven

Compresa la necessità di evitare o ridurre il polling, vediamo come implementare la schedulazione.

La programmazione [Event Driven](#) ad esempio utilizza il sistema, reagendo agli eventi che si manifestano.

In labview questi eventi vengono catturati, dalla "EVENT STRUCTURE", registrando gli eventi a cui vogliamo assegnare un codice di azione.

Il timeout viene visto come un evento perché si scatena sempre come evento di sistema che verifica se il tempo di timeout desiderato è trascorso, restituendo un evento di timeout appunto.

Code

Un'altra tecnica che utilizza la schedulazione, sono le code. In questo caso si implementano con una api, disponibile dalla palette "Queue". Funziona con la funzione "Dequeue" che rimane congelata finché non trova un valore (o messaggio) in coda. Da un'altra parte del codice ci sarà una funzione di "Enqueue" che accoda i messaggi. Il Dequeue schedula tutti i messaggi in coda, se non c'è ne sono rimane in idle, se ho impostato un timeout, trascorso il quale restituisce sulla sua uscita un messaggio vuoto e mette a true il flag di timeout.

Notifier

Funzionano allo stesso modo delle code, utilizzando funzioni di set notifier e Wait For notification.

La differenza è sul fatto che le code hanno memoria della coda, per cui scodano tutti gli elementi cronologicamente messi in coda, senza perdita di dati.

il notifier invece tiene solo un controllo di cambio di stato (notifica) di un elemento, senza memoria, quindi potenzialmente a perdita di dati.

Wait For Panel Activity Function

Abbiamo visto come utilizzando la **EVENT STRUCTURE** si possa limitare le iterazioni solo a quelle effettivamente richieste dagli eventi.

Il [Wait For Front Panel Activity Function](#) aiuta a limitare il codice, in quanto si comporta esattamente come le OCCURRENCE in occasione di un evento del pannello di controllo.

Possiamo usarlo come proposto nell'help di LV nelle finestre di LOGIN o Moduli di raccolta dati che devono attendere la compilazione dei campi prima di eseguire il codice o di terminare la finestra.

Prima si usava mettere tempi di aggiornamento sui 200ms, ma comunque i controlli sulla finestra vengono ridisegnati continuamente inutilmente.

Con questa tecnica si evita lo spreco di risorse CPU e si rende l'applicazione più efficiente.

Ovviamente è sempre da valutare l'utilizzo della struttura a eventi se si vuole un feedback dall'evento generato.



Con GENERATE FRONT PANEL ACTIVITY si può generare l'evento programmaticamente.

Nell'esempio sotto, la string "login" è settata come "update value while typing", ogni carattere inserito esegue un iterazione, perchè è un attività sul pannello, altrimenti rimane in attesa di un evento senza eseguire loop.



Quando la string login contiene la parola "password" viene terminato il while loop.

Conclusione

Possiamo concludere che il polling va limitato e demandato il più possibile al run-time che lo condivide con il sistema operativo.

La struttura a Eventi ne rappresenta la sua implementazione, passando quindi ad una programmazione Event Driven.

Con code e notifier otteniamo una sincronizzazione sul dato, mantenendo una programmazione data-driven.

Altri metodi possono essere utilizzati per mantenere una sincronizzazione su evento piuttosto che interrogare continuamente lo stato dei controlli anche se l'utente non sta interagendo.