Appendice A:DeskLayer Documentazione delle classi

Riferimenti per la struct layWND

`#include <BorderDif.h>`

Membri pubblici

layWND ()
bool **operator==** (**layWND** other)
bool **operator==** (HWND other)
const **layWND** & **operator=** (const **layWND** &other)

Attributi pubblici

HWND **hwnd**
double **weightNSx**
double **weightWEy**

Descrizione Dettagliata

Struttura contenente l'handle di una finestra e il suo relativo peso per i vari bordi.

Documentazione dei costruttori e dei distruttori

layWND::layWND () `[inline]`

Documentazione delle funzioni membro

bool layWND::operator== (layWND *other*) `[inline]`

bool layWND::operator== (HWND *other*) `[inline]`

const layWND& layWND::operator= (const layWND & *other*) `[inline]`

- ### Documentazione dei dati membri

- **HWND layWND::hwnd**

- **double layWND::weightNSx**

- **double layWND::weightWEy**

Riferimenti per la classe BorderDif

`#include <BorderDif.h>`

- ### Membri pubblici

BorderDif ()
void **refreshLayout** ()
void **regulateLayout** (HWND moved)
int **getListPosition** (HWND hwnd, int bord)
void **regLayPosition** (HWND added)
void **regLayPosition** ()
bool **isMoved** (HWND hwnd)
void **pushToCenter** (HWND toPush)
void **moveElement** (HWND **activeWnd**, int oldPos, int newPos)
void **restorePositions** ()
bool **removeElement** (HWND toRemove, int pos, bool complete, bool futuRestore)
int **isInLayout** (HWND toCheck)
std::list< **layWND** > **getWindowsInPos** (int pos)
bool **anchorElement** (HWND hwnd, int **position**)
bool **anchorElement** (HWND hwnd, int **position**, int listPos)
bool **toCenter** (**layWND** toMove, int pos)
void **refreshScreenSize** ()
void **assignWeight** (**layWND** nuova, int pos)
void **ricalcolateWeight** (**layWND** moved, int pos)
void **ridistribuiteWeight** (int pos, double weighToGive)

- ### Attributi privati

std::list< **layWND** > **north**
std::list< **layWND** > **south**
std::list< **layWND** > **east**
std::list< **layWND** > **west**
layWND central
HashMap initPositions
HashMap regPositions
double **centerTempRight**
double **centerTempTop**
double **centerTempBottom**
double **centerTempLeft**
std::list< **layWND** > **wndBuffer**
double **height**
double **width**
double **weightEx**
double **weightWx**

double **weightNy**
double **weightSy**
double **minXweight**
double **minYweight**

- ## Descrizione Dettagliata

Classe che implementa le funzionalità per creare e gestire un layout di tipo border

---

- ## Documentazione dei costruttori e dei distruttori

- **BorderDif::BorderDif ()**

  Costruttore di default.

- ## Documentazione delle funzioni membro

- **void BorderDif::refreshLayout ()**

  Funzione di disegno del layout.

- **void BorderDif::regulateLayout (HWND *moved*)**

  Funzione che regola il layout in conseguenza uno spostamento di una finestra.

- **Parametri:**
  moved Handle della finestra mossa.

- **int BorderDif::getListPosition (HWND *hwnd*, int *bord*)**

  Funzione che restituisce l'indice nella lista di una finestra che si trova in posizione laterale.

- **Parametri:**
  hwnd Handle alla finestra del quale si vuole scoprire l'indice.
  bord Identificatore del bordo nel quale si trova la finestra.

- **Restituisce:**
  L'intero che rappresenta la posizione della finestra nella lista, se la funzione non va a buon fine viene ritornato -1.

- **void BorderDif::regLayPosition (HWND *added*)**

  Viene registrata la posizione di una particolare finestra.

- **Parametri:**
  added Handle della finestra di cui si vuole registrare la posizione.

- **void BorderDif::regLayPosition ()**

  Registra la posizione di tutte le finestre presenti nel layout

- **bool BorderDif::isMoved (HWND *hwnd*)**

Verifica se una finestra è stata mossa.

- **Restituisce:**
  True se la finestra si è mossa, false altrimenti.

- **void BorderDif::pushToCenter (HWND *toPush*)**

Spinge la finestra al centro anche se vi si trovano altre finestre

- **Parametri:**
  toPush Handle della finestra da spingere al centro.

- **void BorderDif::moveElement (HWND *activeWnd*, int *oldPos*, int *newPos*)**

Sposta una finestra da una posizione ad un lato all'interno del layout.

- **Parametri:**
  activeWnd Handel della finestra da muovere.
  oldPos Identificativo del bordo di provenienza della finestra.
  newPos Identificativo del bordo di destinazione della finestra.

- **void BorderDif::restorePositions ()**

Riporta alla posizione precedente all'aggiunta al layout tutte le finestre.

- **bool BorderDif::removeElement (HWND *toRemove*, int *pos*, bool *complete*, bool *futuRestore*)**

Rimuove una finestra dal layout.

- **Parametri:**
  toRemove Handle della finestra da rimuovere dal layout.
  pos Identificativo del bordo dove si trova la finestra da rimuovere.
  complete Se è posto a true indica che tutto ciò che è memorizzato della finestra deve essere rimosso.
  futuRestore Se è posto a true vengono salvati i pesi della finestra nonostante sia rimossa dal layout.
- **Restituisce:**
  True se l'elemento è stato rimosso correttamente, false altrimenti.

- **int BorderDif::isInLayout (HWND *toCheck*)**

Verifica se una finestra fa parte del layout.

- **Parametri:**
  toCheck Handle della finestra da ricercare nel layout.
- **Restituisce:**
  Identificativo del bordo a cui appartiene la finestra, -1 se la finestra non fa parte del layout.

- **std::list< layWND > BorderDif::getWindowsInPos (int *pos*)**

Restituisce tutte le finestre facenti parti di una posizione del layout.

- **Parametri:**
  pos La posizione della quale si vogliono ottenere le finestre

- **Restituisce:**
  Una lista di strutture di tipo **layWND** contenente tutti gli elementi, se ce ne sono, della data posizione.

- **bool BorderDif::anchorElement (HWND *hwnd*, int *position*)**

Aggiunge una finestra al layout.

- **Parametri:**
  hwnd La finestra da aggiungere al layout.
  position La posizione nella quale va aggiunta la finestra.
- **Restituisce:**
  True se l'inserimento è andato a buon fine, false altrimenti.

- **bool BorderDif::anchorElement (HWND *hwnd*, int *position*, int *listPos*)**

Aggiunge una finestra al layout.

- **Parametri:**
  hwnd La finestra da aggiungere al layout.
  position La posizione nella quale va aggiunta la finestra.
  listPos Specifica la posizione all'interno del lato in cui viene aggiunta la finestra.
- **Restituisce:**
  True se l'inserimento è andato a buon fine, false altrimenti.

- **bool BorderDif::toCenter (layWND *toMove*, int *pos*)**

Spsota una finestra presente nel layout al centro.

- **Parametri:**
  toMove Riferimento alla finestra da spostare al centro.
  pos Posizione della finestra da spostare al centro.
- **Restituisce:**
  True se lo spostamento è andato a buon fine, false altrimenti.

- **void BorderDif::refreshScreenSize ()**

Cambia i parametri e ridisegna il layout dopo un evento di cambiamento di risoluzione dello schermo.

- **void BorderDif::assignWeight (layWND *nuova*, int *pos*)**

Assegna un peso ad una finestra che non lo ha.

- **Parametri:**
  nuova Riferimento alla finestra alla quale va settato il peso.
  pos Posizione nella quale la finestra si trova.

- **void BorderDif::ricalcolateWeight (layWND *moved*, int *pos*)**

Regola i pesi in conseguenza di uno spostamento di una finestra.

- **Parametri:**
  moved Riferimento alla finestra spostata.
  pos Nuova posizione della finestra.

- **void BorderDif::ridistribuiteWeight (int *pos*, double *weighToGive*)**

  Ridistribuisce il peso dopo che una finestra ha abbandonato al sua posizione.

- **Parametri:**

  pos Posizione nella quale i pesi vanno ridistribuiti.
  weighToGive Peso della finestra che ha lasciato la posizione che andrà ridistribuito.

Documentazione dei dati membri

- **std::list<layWND> BorderDif::north `[private]`**

  Lista delle finestre nel lato nord.

- **std::list<layWND> BorderDif::south `[private]`**

  Lista delle finestre nel lato sud.

- **std::list<layWND> BorderDif::east `[private]`**

  Lista delle finestre nel lato est.

- **std::list<layWND> BorderDif::west `[private]`**

  Lista delle finesre nel lato ovest.

- **layWND BorderDif::central `[private]`**

  Finestra centrale.

- **HashMap BorderDif::initPositions `[private]`**

  Struttura dati contenente le posizioni delle finestre precedenti all'aggiunta al layout.

- **HashMap BorderDif::regPositions `[private]`**

  Struttura dati contenente la posizione corrente di tutti gli elementi del layout.

- **double BorderDif::centerTempRight `[private]`**

  Coordinate temporanee della cella centrale.

- **double BorderDif::centerTempTop `[private]`**

- **double BorderDif::centerTempBottom `[private]`**

- **double BorderDif::centerTempLeft `[private]`**

- **std::list<layWND> BorderDif::wndBuffer `[private]`**

  Struttura dati contenente handle e pesi delle finestre momentaneamente fuori dal layout.

- **double BorderDif::height `[private]`**
  Altezza del desktop.
- **double BorderDif::width `[private]`**
  Larghezza del desktop.
- **double BorderDif::weightEx `[private]`**
  Peso orizzontale delle finestre ancorate ad est.
- **double BorderDif::weightWx `[private]`**
  Peso orizzontale delle finestre ancorate ad ovest.
- **double BorderDif::weightNy `[private]`**
  Peso verticale delle finestre ancorate a nord.
- **double BorderDif::weightSy `[private]`**
  Peso verticale delle finestre ancorate a sud.
- **double BorderDif::minXweight `[private]`**
- **double BorderDif::minYweight `[private]`**

Riferimenti per la classe GridLayout

`#include <GridLayout.h>`

Membri pubblici

GridLayout ()
void **reset** ()
bool **isInLayout** (HWND toCheck)
bool **isNull** ()
HWND **getElement** (int row, int column)
void **removeElement** (HWND hwnd, bool completely)
bool **anchorElement** (HWND hwnd)
int **getHeight** ()
int **getWidth** ()
int **getRows** ()
int **getColumn** ()
void **drawLayout** ()
void **regulateLayout** (HWND moved, bool recursiveBlock)
bool **checkInterseption** (RECT *moved, RECT *analyzing)
void **changedResolution** ()
void **searchSpace** (HWND moved)
void **restorePositions** ()
bool **isMoved** (HWND hwnd)
void **regLayPosition** ()
void **regLayPosition** (HWND added)
void **revertLayout** ()

- ## Attributi privati

int **rows**
int **columns**
int **height**
int **width**
std::list< HWND > **matrix**
HashMap initPositions
HashMap regPositions
int **curRows**
int **curColumns**

- ## Descrizione Dettagliata

Classe che definisce il classico layout a griglia.

- ## Documentazione dei costruttori e dei distruttori

- ### GridLayout::GridLayout ()

Costruttore di default.

- ## Documentazione delle funzioni membro

- ### void GridLayout::reset ()

Resetta tutti i parametri del layout

- ### bool GridLayout::isInLayout (HWND *toCheck*)

Determina se una finestra fa parte del layout o meno

- **Parametri:**
  toCheck Handle della finestra da ricercare nel layout.
- **Restituisce:**
  True se la finestra è nel layout, false altrimenti.

- ### bool GridLayout::isNull ()

Verifica se il layout è attualmente vuoto

- **Restituisce:**
  True se il layout è vuoto, false altrimenti.

- ### HWND GridLayout::getElement (int *row*, int *column*)

Trova l'handle di una finestra a partire dalle sue coordinate.

- **Parametri:**
  row Riga dell'elemento desiderato.
  column Colonna dell'elemento desiderato.

- **Restituisce:**
  Handle della finestra desiderata.

- **void GridLayout::removeElement (HWND *hwnd*, bool *completely*)**

Rimuove un elemento dall layout.

- **Parametri:**
  hwnd Handle della finestra da rimuovere.
  completely Indica se la finestra deve essere rimossa da tutte le strutture dati del programma.

- **bool GridLayout::anchorElement (HWND *hwnd*)**

Aggiunge una finestra al layout.

- **Parametri:**
  hwnd Handle alla finestra da aggiungere.
- **Restituisce:**
  True se la finestra viene aggiunta correttamente al layout, false altrimenti.

- **int GridLayout::getHeight ()**

Ritorna l'altezza delllo schermo.

- **int GridLayout::getWidth ()**

Ritorna la larghezza dello schermo.

- **int GridLayout::getRows ()**

Ritorna il numero di righe della matrice.

- **int GridLayout::getColumn ()**

Ritorna il numero di colonne della matrice.

- **void GridLayout::drawLayout ()**

Disegna il layout assegnando posizioni e dimensioni alle finestre ricalcolandoli ad ogni chiamata.

- **void GridLayout::regulateLayout (HWND *moved*, bool *recursiveBlock*)**

Regola il layout dopo lo spostamento di una finestra ottimizzandone lo spazio sfruttabile.

- **Parametri:**
  moved Handle della finestra mossa.
  recursiveBock Flag che indica di fermare la ricorsione tra questa funzione e **searchSpace()**.
- **Vedi anche:**
  searchSpace(HWND moved).

- **bool GridLayout::checkInterseption (RECT * *moved*, RECT * *analyzing*)**

  Verifica se tra due rettangoli rappresentanti l'area di una finestra c'è una intersezione e la risolve ridimensionando il rettangolo analizzato.

- **Parametri:**

  moved Puntatore al rettangolo della finestra mossa, questo rettangolo non cambia dimensioni dopo la chiamata a quasta funzione.
  analyzing Puntatore al rettangolo della finestra analizzata, in caso di intersezione viene ridimensionato in modo da eliminarla.

- **Restituisce:**

  True se c'è una intersezione, false altrimenti.

- **void GridLayout::changedResolution ()**

  Adatta il layout in caso di cambio di risoluzione.

- **void GridLayout::searchSpace (HWND *moved*)**

  Ricerca tutto lo spazio disponibile in modo da non lasciare spazi vuoti nel layout ove possibile.

- **Parametri:**

  moved Handle della finestra mossa e quindi che uscirà non modificata dalla chiamata della funzione.

- **void GridLayout::restorePositions ()**

  Ripristina posizioni e dimensioni delle finestre a come erano prima dell'attivazione del layout ove possibile.

- **bool GridLayout::isMoved (HWND *hwnd*)**

  Verifica se una finestra è stata mossa completamente o ridimensionata.

- **Parametri:**

  hwnd Handle della finestra mossa o ridimensionata.

- **Restituisce:**

  True se la finestra si è mossa completamente rispetto alla posizione precedente, false altrimenti.

- **void GridLayout::regLayPosition ()**

  Registra la posizione di tutte le finestre del layout.

- **void GridLayout::regLayPosition (HWND *added*)**

  Registra la posizione della finestra specificata.

- **Parametri:**

  added Handle della finestra di cui registrare la posizione.

- **void GridLayout::revertLayout ()**

  Inverte le posizioni della griglia.

- ## Documentazione dei dati membri

- **int GridLayout::rows `[private]`**

Numero di righe.

- **int GridLayout::columns `[private]`**

Numero di colonne.

- **int GridLayout::height `[private]`**

Altezza dello schermo.

- **int GridLayout::width `[private]`**

Larghezza dello schermo.

- **std::list<HWND> GridLayout::matrix `[private]`**

Lista contenente gli elementi del layout.

- **HashMap GridLayout::initPositions `[private]`**

Struttura per la memorizzazione delle posizioni iniziali delle finestre.

- **HashMap GridLayout::regPositions `[private]`**

Struttura per la memorizzazione delle posizioni attuali delle finestre.

- **int GridLayout::curRows `[private]`**

Numero di righe corrente.

- **int GridLayout::curColumns `[private]`**

Numero di colonne corrente.

Riferimenti per la classe HashMap

`#include <HashMap.h>`

Membri pubblici

HashMap ()
bool **findElement** (HWND key, LPRECT element)
bool **delKeyElement** (HWND key)
bool **insertKeyElement** (HWND key, LPRECT element)
bool **modifyElement** (HWND key, LPRECT newValue)

Attributi privati

std::vector< **WND** > **map**

Descrizione Dettagliata

Struttura dati di tipo **HashMap** basata sulla struttura **WND**.

- **Vedi anche:**
  WND.

Documentazione dei costruttori e dei distruttori

- **HashMap::HashMap ()**

  Costruttore di default.

Documentazione delle funzioni membro

- **bool HashMap::findElement (HWND *key*, LPRECT *element*)**

  Cerca un elemento all'interno della struttura dati.

- **Parametri:**
  key Handle alla finestra che fa da chiave nella tabella hash.
  element Puntatore al rettangolo che risulta essere il valore nella tabella hash.

- **bool HashMap::delKeyElement (HWND *key*)**

  Cancella una coppia chiave-valore dalla struttura.

- **Parametri:**
  key L'handle della finestra da cancellare dalla struttura che fa da chiave per la tabella hash.

- **bool HashMap::insertKeyElement (HWND *key*, LPRECT *element*)**

  Inserisce una coppia chiave-valore nella struttura.

- **Parametri:**

  key Handle di finestra che fa da chiave nella tabella della struttura.
  element Puntatore al rettangolo che rappresenta il valore nella tabella hash.

- **bool HashMap::modifyElement (HWND *key*, LPRECT *newValue*)**

Permette di associare un nuovo valore corrispondente ad una determinata chiave.

- **Parametri:**

  key Handle a finestra e chiave nella tabella della struttura.
  newValue Nuovo valore da associare alla chiave, è un puntatore ad una struttura di tipo RECT

Documentazione dei dati membri

- **std::vector<WND> HashMap::map `[private]`**

Struttura di supporto che contiene i dati della "tabella hash".

Riferimenti per la classe PositionMap

`#include <PositionMap.h>`

Membri pubblici

PositionMap ()
int **findElement** (HWND key)
bool **delKeyElement** (HWND key)
bool **insertKeyElement** (HWND key, int element)
bool **modifyElement** (HWND key, int newValue)

Attributi privati

std::vector< **WNDPOS** > **map**

Descrizione Dettagliata

Struttura dati di tipo HasMap basata sulla struttura dati **WNDPOS**.

Documentazione dei costruttori e dei distruttori

PositionMap::PositionMap ()

Costruttore di default.

Documentazione delle funzioni membro

int PositionMap::findElement (HWND *key*)

Ricerca di un elemento nella struttura in base ad una chiave.

- **Parametri:**
  key Handle di finestra che identifica un valore nella struttura.
- **Restituisce:**
  Un intero che rappresenta la posizione della finestra nel layout.

bool PositionMap::delKeyElement (HWND *key*)

Cancella la coppia chiave-valore identificata da una chiave dalla struttura.

- **Parametri:**
  key Handle della finestra che identifica la coppia chiave-valore da cancellare.
- **Restituisce:**
  True se la coppia è stata cancellata, false altrimenti.

- **bool PositionMap::insertKeyElement (HWND *key*, int *element*)**

Inserisce una coppia chiave-valore.

- **Parametri:**

  key L'handle di finestra che fa da chiave nella struttura.
  element Valore da registrare in coppia con la chiave specificata in key,

- **Restituisce:**

  True se la coppia è stata inserita, false altrimenti.

- **bool PositionMap::modifyElement (HWND *key*, int *newValue*)**

Modifica il valore associato ad una chiave già esistente nella struttura.

- **Parametri:**

  key Handle di finestra al quale si vuole associare un nuovo valore.
  newValue Nuovo valore intero da associare alla chiave.

Documentazione dei dati membri

- **std::vector<WNDPOS> PositionMap::map `[private]`**

Vettore che rappresenta la "tabella hash".

Riferimenti per la classe SettingsManager

`#include <SettingsManager.h>`

Membri pubblici

SettingsManager ()
void **loadOptions** (HWND hwnd)
void **saveOptions** (HWND hwnd)

Documentazione dei costruttori e dei distruttori

SettingsManager::SettingsManager ()

Costruttore di default

Documentazione delle funzioni membro

void SettingsManager::loadOptions (HWND *hwnd*)

Carica tutte le opzioni relative al programma.

- **Parametri:**
  hwnd Handle della finestra del programma.

void SettingsManager::saveOptions (HWND *hwnd*)

Salva tutte le opzioni relative al programma.

- **Parametri:**
  hwnd Handle della finestra del programma.

- ## Riferimenti per la struct WND

`#include <HashMap.h>`

- ## Membri pubblici

bool **operator**== (HWND other)
bool **operator**== (**WND** other)

- ## Attributi pubblici

RECT **rect**
HWND **hwnd**

- ## Descrizione Dettagliata

Struttura di supporto che contiene la rappresentazione di una finestra ovvero il suo handle ed le coordinate del rettangolo occupato.

- ## Documentazione delle funzioni membro

- **bool WND::operator== (HWND *other*) `[inline]`**

- **bool WND::operator== (WND *other*) `[inline]`**

- ## Documentazione dei dati membri

- **RECT WND::rect**

- **HWND WND::hwnd**

- ## Riferimenti per la struct WNDPOS

`#include <PositionMap.h>`

- ### Membri pubblici

bool **operator**== (HWND other)
bool **operator**== (**WNDPOS** other)

- ### Attributi pubblici

int **pos**
HWND **hwnd**

- ### Descrizione Dettagliata

Struttura di supporto che associa un handle di finestra ad un intero che rappresenta una posizione nel layout.

- ### Documentazione delle funzioni membro

- **bool WNDPOS::operator== (HWND *other*) `[inline]`**

- **bool WNDPOS::operator== (WNDPOS *other*) `[inline]`**

- ### Documentazione dei dati membri

- **int WNDPOS::pos**

- **HWND WNDPOS::hwnd**

- ## Riferimenti per msghook.dll

- ### Definizioni

#define **LIBSPEC** __declspec(dllexport)
#define **DECLARE_REGISTERED_MESSAGE**(name, name_s) static
const UINT name = :: RegisterWindowMessage(name_s);

- ### Funzioni

LIBSPEC BOOL **setMyHook** (HWND hWnd)
LIBSPEC BOOL **clearMyHook** (HWND hWnd)

- ## Documentazione delle definizioni

- **#define DECLARE_REGISTERED_MESSAGE(name, name_s) static const UINT name = :: RegisterWindowMessage(name_s);**

Macro per la creazione di messaggi personalizzati

- **#define LIBSPEC __declspec(dllexport)**

- ## Documentazione delle funzioni

- **LIBSPEC BOOL clearMyHook (HWND *hWnd*)**

Disinstalla l'hook dal sistema.

- **Parametri:**
  hWnd Handle della finestra che aveva precedentemente installato l'hook.
- **Restituisce:**
  True se la disinstallazione è riuscita, false altrimenti.

- **LIBSPEC BOOL setMyHook (HWND *hWnd*)**

Funzione che aggancia una finestra al sistema in modo da riceverne informazioni.

- **Parametri:**
  hWnd Handle della finestra che deve ricevere i messaggi intercettati dalla dll.
- **Restituisce:**
  True se l'aggancio al sistema è andato bene, false altrimenti.

- ## Riferimenti per il file principale.cpp

```
#include <windows.h>
#include <psapi.h>
#include <shellapi.h>
#include "BorderLayout.h"
#include "GridLayout.h"
#include "resource.h"
#include "msghook/msghook.h"
#include "SettingsManager.h"
#include "PositionMap.h"
```

- ### Definizioni

#define **_WIN32_IE** 0x0600
#define **NOLAYOUT** 0
#define **BORDER** 1
#define **GRID** 2
#define **WM_TRAY_MESSAGE** (WM_APP+1)
#define **ID_TRAY** 100077
#define **MENU_RESTORE** 1000
#define **MENU_MINIMIZE** 1001
#define **MENU_CLOSE** 1002

- ### Funzioni

DWORD WINAPI **KeysProc** (LPVOID pMyID)
void **setCurrentText** (HWND hwnd)

Funzione che setta nell'area di testo del programma il nome del layout corrente.

bool **isAllowed** (LPSTR exeName)
BOOL **GetExeFromWindow** (HWND hWnd, LPSTR lpszFileName, DWORD dwSize)
BOOL CALLBACK **EnumWindowsProc** (HWND hWnd, LPARAM lParam)
BOOL CALLBACK **DlgProc** (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
int WINAPI **WinMain** (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

- ### Variabili

HWND **prog**
HWND **activeWnd** = NULL
bool **layoutActive** = false
int **activeLayoutType** = NOLAYOUT
BorderLayout * **border**
GridLayout * **grid**
char **buf** [100]
char **buf2** [100]
int **position** = 3
int **temp** = -1
bool **collocated** = false
LPWINDOWPLACEMENT **wndTmp**
int **ThreadNr** = 1
int **index** = 0

bool **zoomed** = false
HKEY **hkey**
char **progPath** [MAX_PATH]
HINSTANCE **hInst**
NOTIFYICONDATA **nid**
HMENU **h_menu**
POINT **pt**
RECT **temporaneo**
int **len**
int **count**
HWND **hList**
SettingsManager setMan
PositionMap minPos

- ## Documentazione delle definizioni

- **#define _WIN32_IE 0x0600**

- **#define BORDER 1**

- **#define GRID 2**

- **#define ID_TRAY 100077**

- **#define MENU_CLOSE 1002**

- **#define MENU_MINIMIZE 1001**

- **#define MENU_RESTORE 1000**

- **#define NOLAYOUT 0**

- **#define WM_TRAY_MESSAGE (WM_APP+1)**

- ## Documentazione delle funzioni

- **BOOL CALLBACK DlgProc (HWND *hwnd*, UINT *message*, WPARAM *wParam*, LPARAM *lParam*)**

  Procedura per il processamento dei messaggi rivolti alla dialog box.

- **Parametri:**

  hwnd Handle della dialog box.
  message Specifica del messaggio.
  wParam Specifica le informazioni addizionali del messaggio.
  lParam Specifica le informazioni addizionali del messaggio.

- **Restituisce:**

  True se il messaggio è stato processato dalla procedura, false altrimenti.

- **BOOL CALLBACK EnumWindowsProc (HWND *hWnd*, LPARAM *lParam*)**

Procedure richiamata dall'API EnumWindows per ogni finestra enumerata, nello specifico aggiunge tutte le finestre aperte al layout automaticamente.

- **Parametri:**

  hWnd Handle della finestra enumerata.
  lParam Eventuali specifiche dell'applicazione chiamante.

- **Restituisce:**

  True per continuare l'enumerazione, false altrimenti.

- **BOOL GetExeFromWindow (HWND *hWnd*, LPSTR *lpszFileName*, DWORD *dwSize*)**

Trova il nome dell'eseguibile che ha generato una finestra.

- **Parametri:**

  hWnd L'handle della finestra del quale si vuole trovare l'eseguibile di origine.
  lpszFileName Stringa di destinazione del nome dell'eseguibile.
  dwSize Spazio che deve occupare la stringa di destinazione.

- **Restituisce:**

  True se viene trovato il nome dell'eseguibile, false altrimenti.

- **bool isAllowed (LPSTR *exeName*)**

Stabilisce in base al filtro sugli eseguibili se il nome dell'eseguibile in input è autorizzato ad essere aggiunto al layout.

- **Parametri:**

  exeName Stringa con il nome dell'eseguibile da testare.

- **Restituisce:**

  True se l'eseguibile è autorizzato, false altrimenti.

- **DWORD WINAPI KeysProc (LPVOID *pMyID*)**

Procedura che definisce le azioni di un thread, nello specifico un thread per la ricezione di inpur system-wide da tastiera.

- **Parametri:**

  pMyID Dati per il thread creato.

- **Restituisce:**

  Il codice di uscita del thread.

- **void setCurrentText (HWND *hwnd*)**

Funzione che setta nell'area di testo del programma il nome del layout corrente.

Appendice B:Schema di comportamento del Border Layout

	Finestra libera	Finestra nel bordo	Finestra centrale
1	Memorizzazione posizione attuale della finestra, e aggiunta nella posizione desiderata.	Spostamento della finestra nella posizione desiderata, adattamento delle dimensioni in base al suo peso.	Spostamento della finestra nella posizione desiderata, adattamento delle dimensioni in base al suo peso.
2	Nessun Effetto	Se le frecce direzionali indicano questa direzione le finestre del bordo verranno ciclicamente scambiate con quella centrale.	La finestra centrale verrà scambiata ciclicamente con le finestre del bordo indicato dalle frecce direzionali.
3	Nessun Effetto	La finestra viene rimossa dal layout, se ne salvano punto di aggancio e pesi. Lo spazio lasciato libero viene assorbito dalle altre finestre dello stesso lato, o se non presenti dai bordi adiacenti o dalla cella centrale.	La finestra viene rimossa da, layout, se ne salvano i pesi e la posizione e lo spazio lasciato rimane vuoto.
4	Nessun Effetto	La finestra viene rimossa dal layout, se ne salvano punto di aggancio e pesi. Lo spazio lasciato libero viene assorbito dalle altre finestre dello stesso lato, o se non presenti dai bordi adiacenti o dalla cella centrale.	La finestra viene rimossa da, layout, se ne salvano i pesi e la posizione e lo spazio lasciato rimane vuoto.
5	Nessun Effetto	La finestra viene ripristinata nell'esatto punto di aggancio che occupava precedentemente se possibile, la dimensione della finestra è data dal rapporto tra il peso attuale della finestra e i pesi delle altre finestre dello stesso bordo.	La finestra viene ripristinata al centro del layout e se nel frattempo la cella è stata occupata da un'altra finestra quest'ultima verrà spostata in una posizione laterale del layout per permettere alla finestra ripristinata di ottenere il proprio posto.
6	Nessun Effetto	La finestra viene rimossa completamente dal layout e nessuna informazione viene salvata dal programma, le altre finestre dello stesso bordo, se presenti, si dividono lo spazio lasciato vuoto, altrimenti lo spazio viene assorbito dalle finestre facenti parte dei bordi adiacenti o dalla cella centrale.	La finestra viene rimossa dal layout, lo spazio lasciato libero rimane vuoto.
7	Nessun Effetto	Se il ridimensionamento avviene in una direzione che interessa solo le celle dello stesso bordo si ha il riadattamento di tutti i pesi in rapporto con la nuova dimensione della finestra spostata, se invece il ridimensionamento interessa altre celle allora tutto il limite del bordo viene riadattato e di conseguenza tutte le finestre delle celle che ne condividevano il confine.	Qualunque movimento della cella centrale comporta il riadattamento delle celle laterali adiacenti al lato o ai lati della cella interessati dal ridimensionamento.Come indicato nelle specifiche della libreria java per il border layout è la cella centrale che da la dimensione a tutte le altre. I pesi e quind le dimensioni delle celle laterali vengono riadattati automanticamente.
8	Nessun Effetto	Se una finestra viene completamente spostata si avrà la rimozione della stesa dal layout con conseguente ridistribuzione dei pesi tra le finestre del bordo se esistenti o altrimenti tra quelle dei bordi adiacenti.	Allo spostamento dellla finestra questa viene completamente rimossa dal layout e lo spazio lasciato al centro rimane vuoto.

Legenda eventi:

1. Pressione ALT+FRECCE.

2. Pressione CTRL+FRECCE.
3. Minimizzazione finestra.
4. Massimizzazione finestra.
5. Ripristino finestra.
6. Chiusura finestra.
7. Ridimensionamento finestra.
8. Movimento finestra.