October, 2008

Using TRIZ for Minimizing Cursor Movements in GUI

Umakant Mishra

Available at: https://works.bepress.com/umakant_mishra/69/
1. Introduction

A graphical user interface is a revolutionary improvement over its predecessor the command line interface. A user can do complex operations just by clicking on icons or dragging and dropping various visual elements on a graphical user interface.

However, a GUI suffers from a typical shortcoming that it requires a lot of pointer movements. As pointer movement is a slow process it affects the overall performance of a GUI operation. Besides too much pointer movement can frustrate a user. Hence, it is desirable to reduce the pointer movements while performing any action through a typical GUI.
1.1 Actions demanding more pointer movement

Although almost every operation in GUI is controlled through the pointer, there are some operations, which require more pointer movements than others. Before attempting to reduce the pointer movements, it is important to find out the operations that require substantial amount of pointer movements.

⇒ Exploring menu items- the user has to move the pointer onto the menu bar to get a drop down, then choose a sub-menu to locate the desired item. If he does not find the desired item, then he has to click on another menu or sub-menu to continue the operation.

⇒ If the user does not remember the mnemonics in a menu he may wrongly navigate to an undesired menu, which may require more keystrokes and navigations to perform an error recovery.

⇒ Selecting a button on the toolbar- the user has to move the pointer from the working area to the toolbar area.

⇒ Exploring a folder button- sometimes a group of similar buttons are kept inside a folder button. The user has to click the folder button to explore the desired button.

⇒ Moving toolbars and other floating objects- to suitable locations on the screen for better visibility and better access.

⇒ Resizing windows- in a multitasking environment the user typically opens multiple applications simultaneously in multiple windows. In such cases the user has to resize windows frequently. Resizing windows requires a lot of mouse movements by holding and dragging the corner points.

⇒ Exploring files- as the files are generally organized in a tree structure, one has to open folders after folders to explore a desired file.

⇒ Scrolling pages- the user has to move the cursor a lot to view different pages of a large document or a large image.

⇒ Moving from one window to another- when the user moves from one window to another, he has to first select the desired window and then navigate to the desired location.
1.2 Problems involved in pointer movement

Pointer movements have some obvious disadvantages, such as,

(always use ⇒ symbol for problem points)

✦ Moving a pointer requires time. Hence operations involving more pointer movements are slower. Although you can increase the speed of a pointer the solution actually worsens the situation.

✦ Pointer movement is a skillful job. Users with shaky hands or having physical challenge cannot move a pointer as accurately as required.

✦ Mistakes in pointer movement may lead to undesirable consequences. More pointer movements thus cause more mistakes.

✦ Typically a pointer device is different than the keyboard. Every time the user starts using the pointer device, he has to move his hands away from the keyboard thus loosing focus from his keyboard activities.

1.3 Ideal Final Result

Ideally the user need not move the pointer to initiate a GUI based operation. In other words the pointer itself should automatically move onto the desired location on the graphical user interface (Ideal Final Result).

Although it is too difficult to manage a GUI without any pointer movement, we should aim at managing GUI operations through minimum amount of pointer movements (Desired Result).

1.4 Methods of reducing pointer movement

The possible areas of reducing pointer movements are as follows.

(always use ⇒ symbol for problem points)

✦ Reducing the distance of menu items from the starting point of the menu can reduce pointer movements. Remote items require more navigation.

✦ The toolbars and toolboxes should be automatically relocated at better locations instead of the user dragging them to relocate again and again.

✦ The desired file should be automatically explored instead of the user opening up folder after folder to explore it.

✦ Windows should resize automatically to ensure best viewing an best utilization of screen space.
Remembering previous status of the window can reduce navigations while switching between windows.

Supplementing the GUI with a Voice Enabled Interface can effectively substitute cursor intensive activities.

Using hot-keys (such as, ^c for copying, ^v for pasting) to substitute pointer intensive GUI operations.

2. Inventions on minimizing cursor movement

2.1 Method of navigating among program menus using a graphical menu tree (US Patent 4821211)

Background problem
Menu items are assigned to do a function, such as displaying help, opening a file or opening another application window. The user has to navigate from one menu to another in the menu tree. Sometimes selecting one menu leads to opening of another menu and the user has to continue navigating in the second menu. If the user fails to memorize the mnemonics of the menu he requires extra navigation and extra keystrokes to perform the error recovery.

Some systems allow program-to-program navigation, but only from a menu already displayed in one window to a previously displayed menu in another window. If the application in the target menu is not open, the user must first open the target window and then navigate through the target menu.

There is a need for a menu navigation technique in which the user need not remember so many complicated menu mnemonics and need not navigate so much in the complicated menu tree.

Solution provided by the invention
Robert Torres invented a menu navigation system (Patent 4821211, assigned by IBM, Apr 89) to move from one menu to another in a hierarchical structure.

The invention facilitates navigation (i) from one menu to another menu in the hierarchy of same application program, or (ii) from one menu of a program to another menu of another program, or (iii) from one menu of a computer to another menu of another computer system.
TRIZ based analysis
The invention facilitates moving from one option of a menu to another option of another menu thus bypassing the number of intermediate steps (Principle-21: Skipping).

In order to activate the target menu, the invention automatically starts the target program (if not already started), automatically opens the target window (if not already opened) and automatically initializes the log-on procedures (if authentication is required) (Principle-25: Self service).

2.2 Command processing system used under graphical user interface utilizing pointing device for selection and display of command with execution of corresponding process (US Patent 5598522)

Background problem
In a conventional command processing system the user has to do a lot of cursor movements with the GUI. For example, in order to delete a file, the user has to go through a pull-down menu-> File-> Delete-> FileOpen-> Select file-> Ok. Although there is only one command applied to an object, the process of selecting that command is very long and has to be repeated every time the command is to be processed.

Solution provided by the invention
Patent 5598522 (invented by Inatomi, assigned by Fujitsu Limited, issued in Jan 1997) disclosed a command processing system, which overcomes the aforesaid difficulties of the prior art.

According to the invention, when the object to be processed is selected, a single command selected from among commands applicable to the selected object is displayed. This method simplifies the process as only a single command is used for the selected object.
Compared to the above conventional process of deleting the file, the steps of the new method are, File open-> Select file-> (the high priority command is automatically displayed) select command-> Ok.

**TRIZ based analysis**
An algorithm decides high priority commands for each different object and displays only that command when that object is selected (Principle-10: Prior action).

The invention reduces the number of steps required to perform a GUI operation (Principle-21: Skipping).

**2.3 Menu control in a graphical user interface (US Patent 5828376)**

**Background problem**
In a graphical user interface, the commands are normally executed from a dropdown menu or from a toolbar. However both of them have their own shortcomings. The menu needs complex navigation before reaching the desired option and the toolbar occupies the real estate of the screen. There is a need to improve the method of accessing and executing commands by overcoming the prior art methods.

**Solution provided by the invention**
Patent 5828376 (Invented by Solimene et al., Assigned to J.D. Edwards World Source Company, Oct 98) provides a “hyperbutton”, a “context sensitive pop-up menu” and a “menu control editor” for easy access to menu options in a GUI.
The hyperbutton displays a user selected default hyperitem from the hyperitems of the menubar. The user can change the default hyperitem using a pop-up menu displayed by using (say) right mouse button. A hyperbutton editor allows the user to configure the hyperbutton and context sensitive pop-up menu be defining the associated hyperitems and their attributes.

The hyperbutton is supposed to be used in addition to the standard menubar and toolbar features of a GUI.

**TRIZ based analysis**

The hyperbutton displays the default hyperitem from the hyperitems of the menubar that helps the user to directly execute from hyperbutton instead of navigating the options through the menu (*Principle-10: Prior Action, Principle-28: Mechanics Substitution*).

The user can change the default hyperitem and configure the hyperbutton through a hyperitem editor (*Principle-15: Dynamize*).

**2.4 Graphical user interface having contextual menus (US Patent 6493006)**

**Background**

Conventionally a menu is displayed at the top of the screen and the user clicks on different icons to get the pull down menus. This process requires a lot of time to navigate through the menu items and found difficult for the beginners to discover the desired item in the menu tree.

**Example Invention:**

Patent 6493006 (Invented by Gourdol et al., assigned to Apple computer, Dec 02) discloses a method of displaying a contextual menu or pop-up menu, that is displayed by an application whenever a user carries out a particular action. The commands which are most frequently used with a particular type of object are displayed in the contextual menu.
The contextual menu appears on the desktop at the location of the cursor, rather than as a pull down menu at the top. The displayed commands are spatially close to the item on which the action is to be performed, which reduces the cursor travel requirements.

TRIZ based analysis:
Ideally the user need not move the pointer to the menu bar; rather the menu should be displayed near the pointer (Ideal Final Result).

The menu should display only those items, which the user needs or are relevant to the context of the user (Ideal Final Result).

The invention proposes a contextual menu which is displayed close to the cursor location and thereby reduces mouse movements (Principle-8: Counterweight).

The contextual menu contains only the items which are relevant for the current operation (Principle-3: Local Quality).

2.5 Voice interaction method for a computer graphical user interface (US Patent 6499015)

Background problem
A typical GUI displays icons, words and other graphical elements on the screen of a video monitor. This method is an improved and efficient alternative to typing commands through a keyboard. But this makes use of excessive and precise mouse movements, which may be difficult for some users. A voice enabled graphical user interface overcomes that difficulty.

Solution provided by the invention
According to the invention, the user speaks a voice command corresponding to the visual components such as menu items, command buttons etc. The computer then associates the voice command to the corresponding display element and does the necessary operation as if the command is entered through a mouse operation.

**TRIZ based analysis**

The voice recognition system reduces cursor movements by substituting GUI buttons with voice commands (**Principle-28: Mechanics substitution**).

When a voice command identifies an element on the GUI display the computer displays a menu of voice commands associated with that element (**Principle-23: Feedback**).

### 3. Summary

The need of pointer movement is an obvious disadvantage of a GUI. It is necessary to reduce pointer movements for any GUI based operation.

We can avoid pointer movements by using voice interfaces, keyboard (command) interfaces, gesture interfaces and other type of interfaces, but that will loose the benefits of a graphical user interface. Ideally we should have a GUI that requires minimum pointer movements.

There are many inventions on reducing the pointer movements in a GUI. Some methods reduce the distance to access the remote menu items, some methods automate the process of resizing and relocating windows and toolboxes, some methods bring the desired menu or command near to the cursor location instead of needing the cursor to move to the menu or command location.
Reference to patents:


Other references:


