June 2023

ENABLING DESKTOP SWITCHING DURING SCREEN SHARING BASED ON CURSOR MOVEMENT

Deepesh Arora

Rajarshee Dhar

Ram Mohan R

ENABLING DESKTOP SWITCHING DURING SCREEN SHARING BASED ON CURSOR MOVEMENT

AUTHORS:
Deepesh Arora
Rajarshee Dhar
Ram Mohan R

ABSTRACT

Existing screen sharing-related solutions primarily focus on the sharing of a single screen or desktop. For example, to share a screen a user typically has to select the screens that they wish to share every time they move to a different screen. This can break a collaboration flow if a user has multiple screens or desktops, and they keep moving across them during a screen sharing event. Techniques are presented herein that fill this gap by allowing a presenter to share their desktops based on the current location of their mouse. As the mouse moves from one desktop to another, a shared desktop may also change correspondingly.

DETAILED DESCRIPTION

All of the existing screen sharing-related solutions focus principally on the sharing of a single screen or desktop. Currently, a user must select the screens that they wish to share every time they move to a different screen. This can break a collaboration flow if a user has multiple screens or desktops, and they keep moving across them during a screen sharing event.

As a result, there is a need to create a seamless screen sharing flow when a presenter has multiple screens and the move across those screens. However, existing attempts at providing such a solution suffer from a significant issue. Consider the case where a user is working across multiple desktop screens. If the user wishes to share their entire work area, irrespective of a screen (e.g., if the user is currently working on Screen 1 and they later move their cursor to Screen 2), then the only way that the user can share that screen is for them to stop the currently-shared screen and start sharing again with Screen 2. Further, if the user has applications that are spread across multiple screens, then they would need to reshare every time they switch their screen, which today is a very annoying experience. To obviate such a need, a user is forced to bring all of their applications under a single screen or desktop that they wish to share, creating a suboptimal experience.

To address such issues, techniques are presented herein that provide a novel mechanism for tracking a mouse cursor in order to facilitate screen sharing based on the mouse cursor tracking. Under such an approach, whichever screen has the mouse cursor, out of multiple screens or desktops that a user is working on, that particular screen will be considered the active screen and that screen will be shared to other users. As soon as the user moves their mouse cursor from one screen to another, the underlying shared screen would also change without the user needing to worry about which screen had been previously shared or where she is currently working. Further aspects of the presented techniques support an application-based sharing approach, instead of a desktop-based sharing approach, which may also be based on the movement of a mouse cursor.

Use of the techniques presented herein offers a number of advantages. For example, under aspects of the presented techniques, a user does not need to be concerned about selecting, before sharing begins, multiple applications that are to be shared. The user may just move their mouse cursor to a particular application, and it will be shared.

The presented techniques operate from a presenter's perspective, not from an audience perspective. An audience will always have only one screen in their stage, and they will not be confused by multiple screen areas that may be selected. Under the presented techniques, it is the presenter's responsibility to determine what she wants to share and that will be wherever her cursor goes.

Consider an example flow through which techniques of this proposal may be illustrated. Under the example, which takes an end user's perspective, a user is working on three screens. A first screen is a native laptop that is running an application, a second screen has a browser, and a third screen has a terminal and a code editor that has been opened. When starting a screen share, the user may select a mode as either mouse tracking-based desktop sharing (according to a first aspect of the presented techniques) or mouse tracking-based application sharing (according to a second aspect of the presented techniques).

A first aspect of the techniques presented herein encompasses a mouse tracking-based desktop sharing flow. Referring to the illustrative example that was introduced above, after starting a screen share an audience may see the presenter's Screen 2, as shown in Figure 1, below.

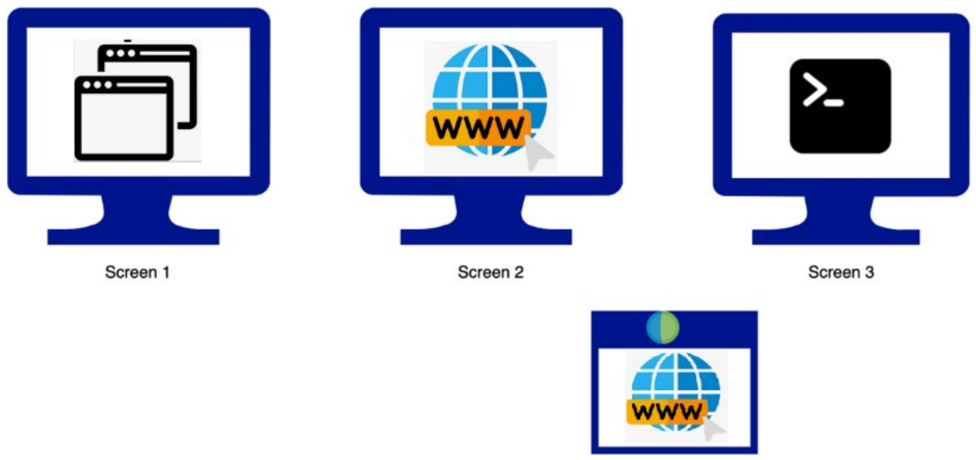

Figure 1: Exemplary Screen Share – Initial

As soon as the user (i.e., the presenter) moves their cursor from Screen 2 to Screen 3, an online meeting client would change the underlying screen buffer or frame buffer, which is being streamed and now, instead of Screen 2, Screen 3 will be sent over the network to the audience. Figure 2, below, depicts this activity.

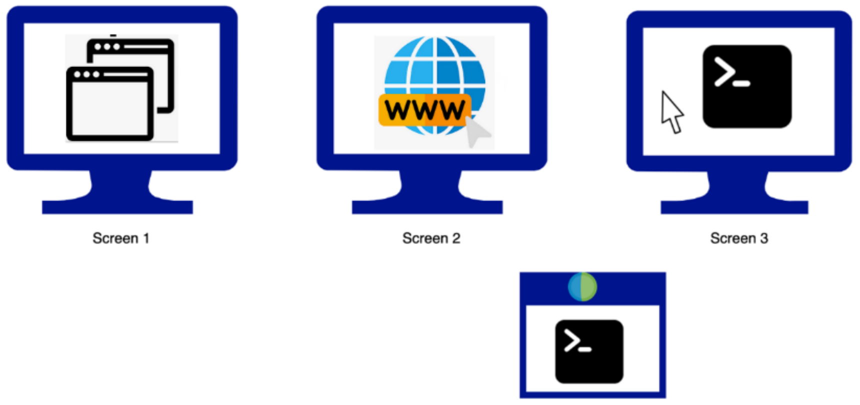

Figure 2: Exemplary Screen Share – Changed

The online meeting client may initially share the monitor where the online meeting application resides and the same would also have a mouse cursor. To identify the current area that is being shared, it is possible to obtain the current coordinates of a mouse cursor through a call to the Microsoft Windows Win32 application programming interface (API) function `GetCursorPos`. That function call will return a structure of type `LPPOINT` which contains the x- and y- coordinates of the cursor’s location.

Using the above-described x- and y- coordinates, it is possible to obtain the current monitor’s handle "`HMONITOR`" by passing a "`POINT`" structure as an input argument to the API function "`MonitorFromPoint.`" The `HMONITOR` handle may then be used to retrieve all of the information about that monitor through a call to the API function "`GetMonitorInfoA`" that will return a pointer to a "`MONITORINFO`" structure.

The above-described sequence of function calls will provide data:

```
typedef struct tagMONITORINFO {
  DWORD cbSize;
  RECT  rcMonitor;
  RECT  rcWork;
  DWORD dwFlags;
} MONITORINFO, *LPMONITORINFO;
```

from which it is possible to obtain a "`RECT`" data structure for the monitor and its working area.

To change a shared region, it is necessary to pass the new coordinates (which may be extracted from the above-described `MONITORINFO` structure) to the API method "`IRDPSRAPISharingSession::SetDesktopSharedRect`."

It is not necessary to repeat the above-described steps every time that a user moves the cursor. Rather, a location-based map (location-map) may be maintained which may directly return the "`RECT`" data structure of the monitor's working area when x- and y- coordinates representing the mouse's current location are provided as an input.

To determine whether a mouse cursor has been moved from one monitor to another, it is possible to register to receive a notification regarding mouse callback events through a "`WM_MOUSEMOVE`" message. Using the x- and y- coordinates that were returned by the "`GetCursorPos`" function call that was described above, the location-based map (location-map) may be searched. If an entry is found, then the coordinates in the located "`RECT`" data structure may be used to change the shared area by calling the API method "`IRDPSRAPISharingSession::SetDesktopSharedRect`." If an entry is not found in the map, then the above-described sequence of function calls may be repeated.

As a result of the above activity, the sharing transition would be so smooth that it would not break a presenter's flow (which would be broken during a re-sharing operation).

Under aspects of the techniques presented herein, a button may also be provided to a user to switch sharing if the user does not want automatic switching on the basis of cursor movement. That button may only be enabled only for the desktop or application where the cursor will be. For example, if a user moves from Screen 1 to Screen 2 (as depicted in Figure 1, above) then such a button may appear only on Screen 2 (to switch sharing) and as soon as the user returns to Screen 1 (which is currently being shared) that button may disappear.

In support of a button as described above, aspects of the techniques presented herein support the integration of a lightweight artificial intelligence (AI) model to learn user behavior or patterns, which would help an online meeting client to take a decision on when and when not to switch desktop sharing. Such user behavior may encompass how long a user stays on a desktop, whether a user is employing a stop button (as described above) to not switch sharing, how fast a user switches across the different desktops, etc.

The techniques presented herein encompass a number of novel features. Those features include the dynamic selection of a screen buffer, out of multiple available buffers (one for each external display), for sharing based on the location of a mouse cursor; the real-time pointing of an application media stream to different screens (i.e., a change in a shared region); the tracking of a mouse cursor location through the use of callback events from an operating system (OS), for changes in a cursor location with respect to multiple desktops or applications, to change the active shared region; the mapping of a mouse cursor with a screen area and the storage of a reference to the underlying screen buffers (e.g., an "`HMONITOR`"type structure) which may be used to switch sharing; a mechanism for detecting an active desktop (i.e., decision making regarding which desktop is to be shared); and a defensive mechanism to avoid switching between screens if a user is moving across different screens too quickly.

As described and illustrated in the above narrative, while existing solutions may support the sharing of either multiple applications or an active application, the techniques presented herein identify an active screen or desktop out of multiple external desktops based on mouse cursor tracking. Aspects of the presented techniques focus on the switching of an active shared desktop on the basis of the current location of a mouse cursor.

The key highlights of the presented techniques include the processing of events related to a change in cursor location from one desktop to another, the handling of external desktops, the identification of the coordinates of a desktop, and the mapping of a coordinate range to desktop handlers.

Additionally, the presented techniques may include a number of novel aspects. For example, the techniques may facilitate an accumulation of metadata that can be used to store information related to the coordinates that are covered by each desktop. That metadata may be stored in such a way that at any given point in time if the current location of a mouse cursor is known, it is possible to obtain the coordinates of the underlying desktop by using direct hashing (i.e., in constant time, or O(1) in Big O notation). Further, mechanism can be provided to facilitate the real-time switching of an active desktop that is being shared without the user needing to manually perform such a switch.

Moreover, techniques presented herein may facilitate the prevention of automatic desktop switching as a defensive approach if a user is moving their cursor across multiple desktops too fast. Patterns may be learned from the user's activities and appropriate actions may be taken accordingly. Still further, techniques presented herein may provide for tracking a mouse cursor's location through the use of callback events from an OS, for changes in a cursor location with respect to multiple desktops, in support of changing an active shared region. Additionally, techniques herein may facilitate learning a user's activity across all of the desktops, which can improve decision making regarding the switching of sharing.

In summary, techniques presented herein allow a presenter to share desktops based on a current mouse location of the presenter. As the presenter's mouse moves from one desktop to another, a shared desktop may also change accordingly.