July 2022

SCROLLING AND RESIZING-AWARE ANNOTATION FEATURE FOR COMMUNICATION AND COLLABORATION PLATFORMS

Deepesh Arora

Sreekanth Narayanan

Priya Kesari

Keerthana S

SCROLLING AND RESIZING-AWARE ANNOTATION FEATURE FOR COMMUNICATION AND COLLABORATION PLATFORMS

AUTHORS:
Deepesh Arora
Sreekanth Narayanan
Priya Kesari
Keerthana S

ABSTRACT

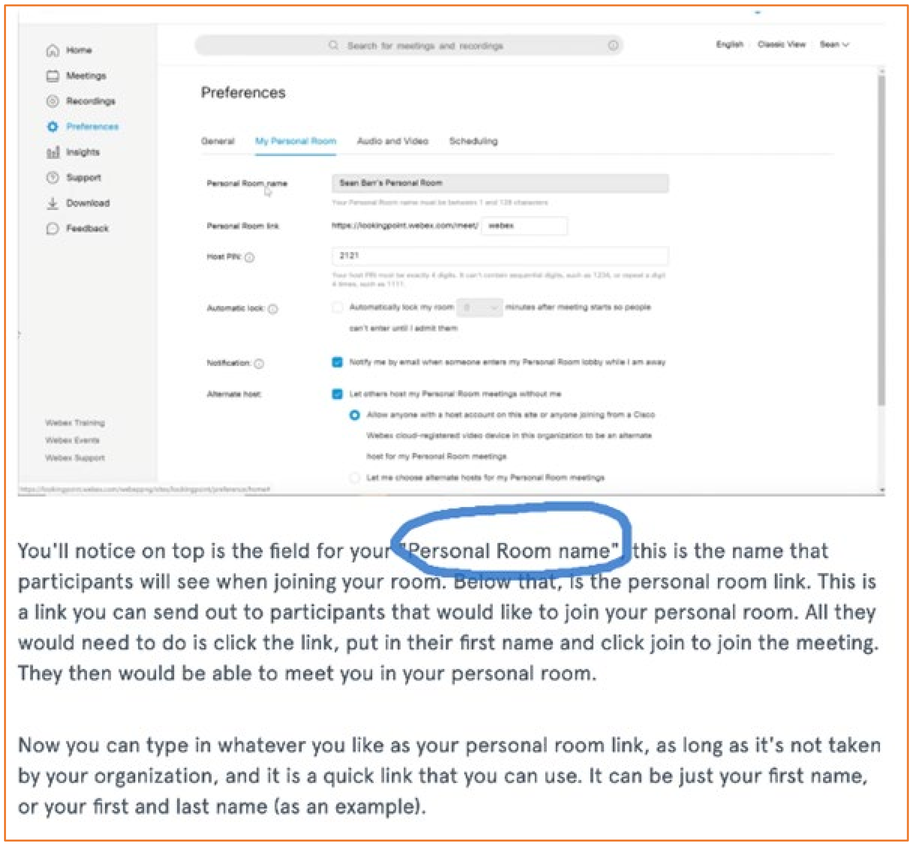
In communication and collaboration platforms (which may support, among other things, video conferencing, online meetings, screen sharing, webinars, etc.), annotation is a critical feature. That feature is playing a very important role during the Coronavirus Disease 2019 (COVID-19) period where, for example, universities and companies are relying heavily upon online meeting experiences to seamlessly deliver classes, sessions, presentations, etc. However, current annotation capabilities are not able to adapt to the dynamic nature of a screen share. To address such a challenge, techniques are presented herein that enhance scrolling within or the resizing of an underlying application. Aspects of the presented techniques encompass the binding of a scroll position to drawn annotations, the establishment of a relationship between annotations and application size, the automatic adjustment of pixel density following the resizing of an application, and the normalizing of a pixel's density.

DETAILED DESCRIPTION

In communication and collaboration platforms (which may support, among other things, video conferencing, online meetings, screen sharing, webinars, etc.), annotation is a critical feature that is playing an important role during the Coronavirus Disease 2019 (COVID-19) period during which many universities and companies are relying heavily upon online meeting experiences with, respectively, students and clients or employees.

However, a problem exists with current annotation capabilities. Consider a scenario in which a university wishes to offer remote online education to its students during the COVID-19 period. In this scenario, consider that a professor “Bob” is a lecturer at the university and is looking forward to delivering online lectures to students who physically are in remote locations. During one of his lectures, he teaches his students how to use a communication and collaboration platform and while doing so he believes using annotations will help the students to better understand.

In a first scenario under the instant hypothetical example, professor Bob highlights the phrase "Personal Room name" in a line of text as illustrated in Figure 1, below.



Preferences

Personal Room name: Sean Barr's Personal Room

Personal Room link: https://lookingpoint.webex.com/meet/ webex

Host PIN: 2121

Automatic lock: Automatically lock my room minutes after meeting starts so people can't enter until I admit them

Notification: ✓ Notify me by email when someone enters my Personal Room lobby while I am away

Alternate host: ✓ Let others host my Personal Room meetings without me

● Allow anyone with a host account on this site or anyone joining from a Cisco Webex cloud-registered video device in this organization to be an alternate host for my Personal Room meetings

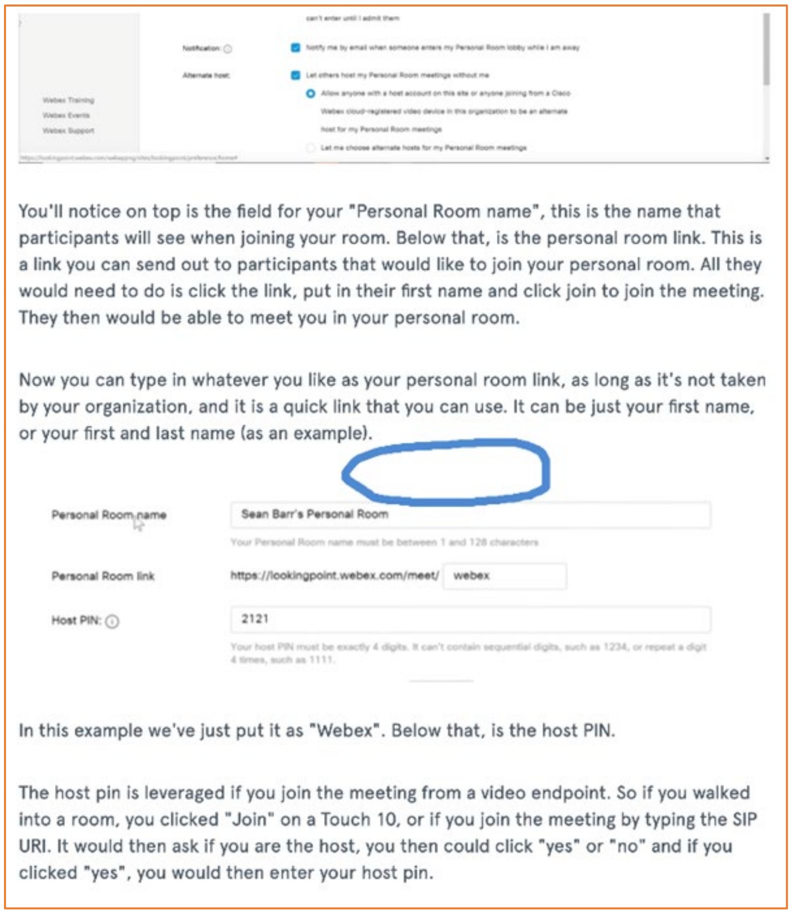
○ Let me choose alternate hosts for my Personal Room meetings

You'll notice on top is the field for your "Personal Room name", this is the name that participants will see when joining your room. Below that, is the personal room link. This is a link you can send out to participants that would like to join your personal room. All they would need to do is click the link, put in their first name and click join to join the meeting. They then would be able to meet you in your personal room.

Now you can type in whatever you like as your personal room link, as long as it's not taken by your organization, and it is a quick link that you can use. It can be just your first name, or your first and last name (as an example).

Figure 1: Exemplary Annotation

After the professor completes the above-described annotation, he scrolls up on the website. Following that scrolling, the professor is disappointed to see that his annotation (as depicted in Figure 1, above) is no longer located over the text "Personal Room Name." Instead, the highlighted portion is now pointing to an empty space, as depicted in Figure 2, below, resulting in unnecessary confusion for his students.



can't enter until I admit them

Notification: ☑ Notify me by email when someone enters my Personal Room lobby while I am away

Alternate host: ☑ Let others host my Personal Room meetings without me

● Allow anyone with a host account on this site or anyone joining from a Cisco Webex cloud-registered video device in this organization to be an alternate host for my Personal Room meetings

○ Let me choose alternate hosts for my Personal Room meetings

Webex Training
Webex Events
Webex Support

You'll notice on top is the field for your "Personal Room name", this is the name that participants will see when joining your room. Below that, is the personal room link. This is a link you can send out to participants that would like to join your personal room. All they would need to do is click the link, put in their first name and click join to join the meeting. They then would be able to meet you in your personal room.

Now you can type in whatever you like as your personal room link, as long as it's not taken by your organization, and it is a quick link that you can use. It can be just your first name, or your first and last name (as an example).

Personal Room name: Sean Barr's Personal Room
Your Personal Room name must be between 1 and 128 characters

Personal Room link: https://lookingpoint.webex.com/meet/ webex

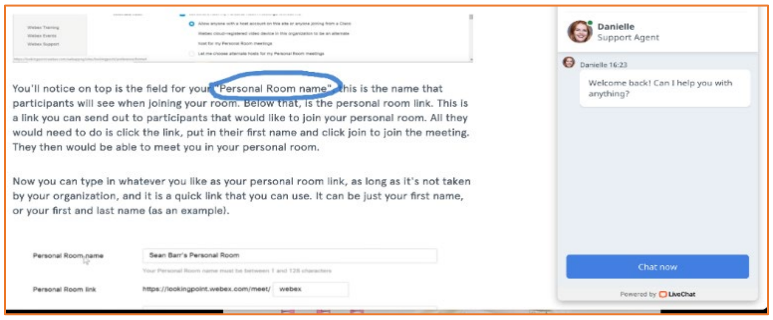
Host PIN: 2121
Your host PIN must be exactly 4 digits. It can't contain sequential digits, such as 1234, or repeat a digit 4 times, such as 1111.

In this example we've just put it as "Webex". Below that, is the host PIN.

The host pin is leveraged if you join the meeting from a video endpoint. So if you walked into a room, you clicked "Join" on a Touch 10, or if you join the meeting by typing the SIP URI. It would then ask if you are the host, you then could click "yes" or "no" and if you clicked "yes", you would then enter your host pin.

Figure 2: Exemplary Annotation After Scrolling

In a second scenario under the instant hypothetical example, professor Bob shares a browser window with his students and, for any number of reasons, he wishes to open another application on his desktop for which he must change the size of the browser window to create additional room for the other application. Figure 3, below, illustrates elements of an exemplary desktop before the browser window is resized.



You'll notice on top is the field for your "Personal Room name", this is the name that participants will see when joining your room. Below that, is the personal room link. This is a link you can send out to participants that would like to join your personal room. All they would need to do is click the link, put in their first name and click join to join the meeting. They then would be able to meet you in your personal room.

Now you can type in whatever you like as your personal room link, as long as it's not taken by your organization, and it is a quick link that you can use. It can be just your first name, or your first and last name (as an example).

Personal Room name: Sean Barr's Personal Room

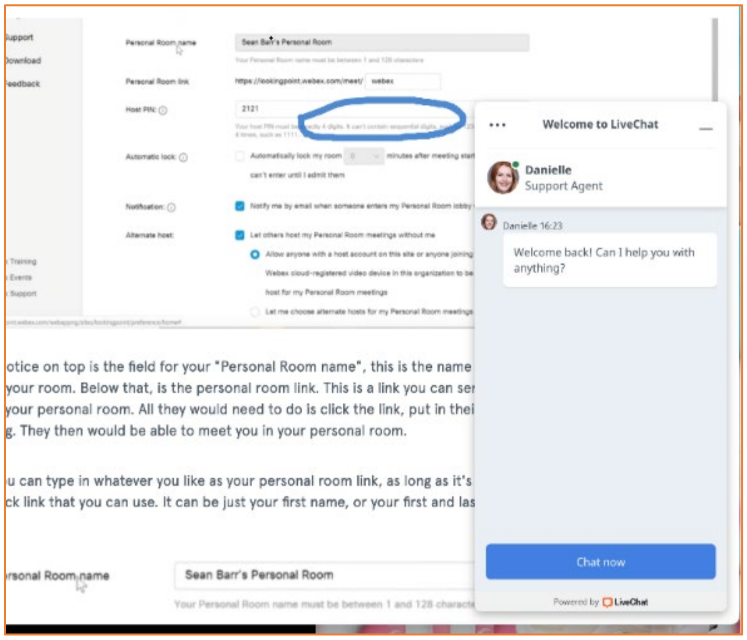
Personal Room link: https://lookingpoint.webex.com/meet/ webex

Danielle
Support Agent

Danielle 16:23
Welcome back! Can I help you with anything?

Figure 3: Exemplary Desktop Before Window Resizing

Figure 4, below, illustrates elements of the exemplary desktop from Figure 3, above, after the browser window is resized.



Personal Room name: Sean Barr's Personal Room

Your Personal Room name must be between 1 and 128 characters

Personal Room link: https://lookingpoint.webex.com/meet/ webex

Host PIN: 2121

Your host PIN must be exactly 4 digits. It can't contain sequential digits, 4 times, such as 1111.

Automatic lock: Automatically lock my room 0 minutes after meeting star can't enter until I admit them

Notification: Notify me by email when someone enters my Personal Room lobby

Alternate host: Let others host my Personal Room meetings without me

Allow anyone with a host account on this site or anyone joining Webex cloud-registered video device in this organization to be host for my Personal Room meetings

Let me choose alternate hosts for my Personal Room meetings

Welcome back! Can I help you with anything?

otice on top is the field for your "Personal Room name", this is the name your room. Below that, is the personal room link. This is a link you can ser your personal room. All they would need to do is click the link, put in thei g. They then would be able to meet you in your personal room.

u can type in whatever you like as your personal room link, as long as it's ck link that you can use. It can be just your first name, or your first and las

rsonal Room name: Sean Barr's Personal Room

Your Personal Room name must be between 1 and 128 characte

Figure 4: Exemplary Desktop After Window Resizing

Following his resizing of the browser window, professor Bob observes that the annotation that he made earlier (as depicted in Figure 3, above) is no longer at the correct location. As shown in Figure 4, above, the highlighted section is now located at a text box instead of over the text "Personal Room Name" which, again, has the potential to confuse his students.

As described and illustrated above, it is clear that current annotation capabilities are not able to adapt to the dynamic nature of a screen share. Techniques are presented herein, which will be described and illustrated in the below narrative, that address such a problem. Aspects of the presented techniques build upon certain existing capabilities and make them more robust.

The next portion of the instant narrative describes and illustrates how aspects of the techniques presented herein may be applied within a Microsoft Windows (i.e., Windows) operating system (OS) to address the scrolling challenge under the first scenario that was detailed above.

In any Windows application which supports scrolling (whether horizontal, vertical, or both), `WM_VSCROLL` and `WM_HSROLL` messages are sent to the application when a scroll event occurs in, respectively, the window's standard vertical and horizontal scrollbar. While processing such events, the application may call the Windows function `GetScrollInfo` to retrieve further information about the scrolling.

Through the Windows-supplied application programming interface (API), the `GetScrollInfo` function accepts three input parameters:

- `hwnd`. A handle to the window (e.g., a browser window in the instant first scenario) for which information is to be gathered. Note that it is possible to easily iterate through all of the open applications in a system and provide the appropriate handle to this API call.
- `nBar`. The constant `SB_CTL` for controlling the scroll bar, the constant `SB_HORZ` for obtaining the parameters for the horizontal scroll bar of the window, or the constant `SB_VERT` for obtaining the parameters for the vertical scroll bar of the window.
- `lpsi`. A pointer to a `SCROLLINFO` structure (as described below).

The `SCROLLINFO` structure contains the following elements:

```
{
UINT cbSize;
UINT fMask;
int nMin;
int nMax;
UINT nPage;
int nPos;
int nTrackPos;
} SCROLLINFO, *LPSCROLLINFO;
```

where a number of those elements are of interest and note, including:

- `fMask`. This value may be the constant `SIF_TRACKPOS` in which case the `nTrackPos` member contains the current position of the scroll box while the user is dragging it.
- `nMin`. The minimum scrolling position.
- `nMax`. The maximum scrolling position.
- `nPage`. The page size in device units.

- `nTrackPos`. The immediate position of the scroll box that the user is dragging. This value provides a real-time position of the scroll box.

As described above, a `SCROLLINFO` structure may be populated through a call to the `GetScrollInfo` function.

Figure 5, below, presents the anatomy of an exemplary scrollbar highlighting various of the structure elements that were described above.

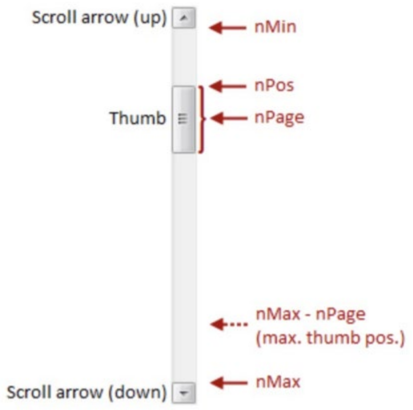


Figure 5: Anatomy of a Scrollbar

Aspects of the techniques presented herein leverage all of the scrolling-related data that was described above to dynamically adjust the positions of any annotations that may appear on a screen (so as to address the first scenario that was detailed above). The steps that are associated with such dynamic adjustments are described below.

Under a first step, the value of the structure element `nTrackPos` contains the position of the thumb when it is currently being dragged to a new position. This value is read-only and it cannot be directly changed programmatically. It is possible to leverage `nTrackPos` to track the movement of the scroll bar, since for every scroll event `WM_VSCROLL` or `WM_HSCROLL` messages we will be received. Inside of those messages it is possible to keep track of the thumb position as a user drags it.

During a second step, the previous position and the current position of a scroll bar may be tracked so that it is possible to determine the delta by which it has been moved either horizontally or vertically. Then, at a third step, on every scroll event (as described above) a delta value may be determined.

Under a fourth step, annotation data may be stored in the form of pixels where each pixel corresponds to the (x, y) coordinates relative to the entire screen. Consequently, a delta value (as described above) will also be in form of an (x, y) pair where a change in just an x value will indicate that the screen has been moved horizontally and, similarly, a change in just a y value will indicate that the screen has been moved vertically.

During a fifth step, by using all of the metadata the annotation figures may be shifted to their new position. For example, consider a single pixel whose initial position is the pair (X: 35, Y:150) and the delta value that was retrieved from handling scroll events is the pair (dX: 0, dY: 50). The delta value indicates that scrolling was done vertically and, since the value dY is greater than zero, the scroll bar moved downwards (i.e., the content moved upwards). Therefore, the annotation pixels also need to be shifted upwards as they need to be kept intact with the underlying region of the application where they were drawn. The new coordinates of the instant pixel thus become (X1: 35, Y1: 150 - 50 = 100) or the (x, y) coordinate pair (35, 100).

For simplicity of exposition, the techniques presented herein were described and illustrated in the above narrative in connection with a Windows-based solution to the previously-described first scenario. However, it is important to note that the presented techniques are equally applicable to other OSs. For example, helper APIs are available under the Apple macOS OS and offer the function `NSScrollView` as an equivalent to the Windows function `GetScrollInfo`.

The next portion of the instant narrative describes how aspects of the techniques presented herein may be applied within a Windows OS to address the resizing challenge under the second scenario that was detailed above. The steps that are associated with such activity are described below.

Under a first step, the Windows-supplied API offers the function `GetWindowRect` to which may be passed the handle of the application which is being queried. A call to that function will return a `RECT` structure that receives the screen coordinates of the upper-left hand and the lower-right hand corners of the window.

During a second step, the changes in a window's size may be obtained through either polling or the registration of certain callback events (such as with scrolling and receipt of `WM_VSCROLL` and `WM_HSROLL` messages).

Under a third step, for two consecutive events it is possible to calculate the delta value for both of the coordinates of an application's rectangular frame (i.e., the upper-left hand and the lower-right hand corners). By processing those delta values, it is possible to estimate by how much margin or proportion the size of a window has become either smaller or larger.

A fourth step leverages the fact that an annotation is simply a collection of pixels. The particulars of this step may be understood through an illustrative example.

Consider the change in position of a single pixel on an event as described above for the TextPad application where the initial positions that were returned in a `RECT` structure are (left: 10, top: 800, right: 600, bottom: 75) and the positions that were returned in a `RECT` structure following resizing are (left: 10, top: 800, right: 800, bottom: 125). Assume that the initial position of an annotation pixel that is under consideration in this example is (120, 330) – i.e., somewhere within the rectangle of that application. The returned values indicate that the application was (1) expanded horizontally but (2) shrunk vertically. The handling of each of those two cases will be described below.

A first case encompasses horizontal expansion. As indicated above, the x-coordinates changed from (L:10, R:600) to (L:10, R:800). This indicates that the expansion was towards the right by a proportion of 4/3 (i.e., 800/600). As noted previously, the x-coordinate of the pixel that is under consideration is 120.

Since the area of the application has increased, that change must be accounted for by having more pixels within the same area. The density of the pixels (i.e., the number of pixels per inch (PPI)) needs to be kept in mind while considering such a change. Modern systems generally have high PPI values. Therefore, through a proportionality rule (which will be described below) the density of the pixels may also be automatically adjusted. Despite the pixels moving away from each other, the PPI is large enough for the distance to be unobservable. However, in the case of an expansion that results in the pixels being too far apart (yielding a distorted figure) extrapolation may be leveraged to fill the void between the pixels.

Since it is necessary to keep the density of the pixels proportionate with the new area that is covered by the application, this requires finding the new spot for the pixels using the ratio of the change. The following formula may be leveraged to calculate such a new position:

(old X Position of Pixel - Left value of RECT) /
(old Right value of RECT - old X Position of Pixel) =
(new X Position of Pixel - Left value of RECT) /
(new Right value of RECT - new X Position of Pixel)

The above approach maintains the density of the pixels proportionate to the expanded area. In the instant example, the expansion ratio is approximately 1.33. This implies that if 10 pixels were previously covering 3 units of an area, those pixels should now cover 4 units of the area. Through the techniques presented herein, as described above, the new position (X) of the pixel that is under consideration may be calculated as (120-10)/(600-120) = (X-10)/(800-X) resulting in X being equal to 157.

A second case encompasses vertical contraction. As indicated above, the y-coordinates changed from (B:75, T:800) to (B:125, T:800). This indicates that the contraction was downwards by a proportion of 3/5 (i.e., 75/125). Consequently, the pixel that is under consideration (which, as noted above, has a y-coordinate of 330) should move upwards. The following formula may be leveraged to calculate such a new position:

(new Y Position of Pixel - New Bottom value of RECT) /
(Old Top value of RECT - new Y Position of Pixel) =
(Old Y Position of Pixel - Old Bottom value of RECT) /
(Old Top value of RECT - old Y Position of Pixel)

Through the techniques presented herein, as described above, the new position (Y) of the pixel that is under consideration may be calculated as (Y - 125)/(800 - Y) = (330 - 75)/(800 - 330) resulting in Y being equal to 362.

By combining all of the above, the fourth step may develop a new position for the collection of pixels that encompass an annotation. In the instant example, the new position of the pixel that is under consideration is the (x, y) coordinate pair (157, 362).

The techniques presented herein offer a number of innovative capabilities or features. A first feature encompasses the binding of a scroll position to drawn annotations. Once the relationship between annotation pixels and a scroll position is established, those artifacts will always be synchronized and thus will provide a much smoother experience to an end user. A second feature encompasses the establishment of a relationship between annotations and application size. This allows a user to have an enhanced experience while drawing annotations and resizing windows without bothering with any distortion of the drawn annotations.

A third feature encompasses the automatic adjustment of pixel density following the resizing of an application. Through a proportionality rule (as described above), pixels will either move away from each other (making a figure look bigger) or they will overlap each other (making the figure look smaller). Further, a fourth feature encompasses the normalizing of a pixel's density. If pixels are too far apart, new pixels may be added between such pixels using extrapolation.

In summary, techniques have been presented that enhance scrolling within or the resizing of an underlying application. Aspects of the presented techniques encompass the binding of a scroll position to drawn annotations, the establishment of a relationship between annotations and application size, the automatic adjustment of pixel density following the resizing of an application, and the normalizing of a pixel's density.