MemoTube: Transient Note-taking in a Video Player

Evangelia Eirini Koktsidou 118884
Hiyeon Kim 118654

CONCEPT

MemoTube is a video player app that allows users to take notes while they are watching a video. We applied the idea of using additional contact points to introduce transient interaction from the paper[1]. The difference is that the transient interaction in MemoTube is note-taking(memo), not the gestures. The idea was applied as follows.

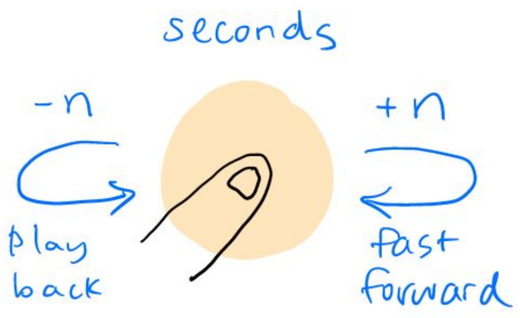

Figure 1: One-Finger mode

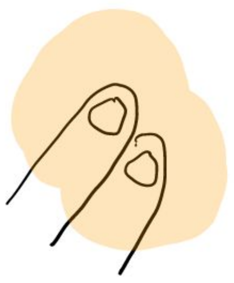
Figure 2: Two-Finger mode

After the user chooses what to play, the video starts playing in a normal video player view. When the user puts the non-dominant finger(s) as additional contact point(s) on the screen, note-taking functionality is enabled as a quasi-mode. This means that the user can take memos as long as the contact point(s) is placed. The canvas will be semi-transparent, layered on top of the video. This way, the user can see the content underneath while taking the memo.
We came up with two options for note-taking mode; One-Finger (Figure 1) or Two-Finger mode (Figure 2). The main difference between the two is whether or not the video is paused or not. In One-Finger mode, the memo is taken on top of the playing video, whereas in Two-Finger mode, the video will simply be paused when the note-taking mode is on. On top of that, the One-Finger mode allows users to jump in the video through swiping quickly the finger that is being used as the contact point. Swiping left and back rewinds the N seconds, while right and back fast forwards N seconds.

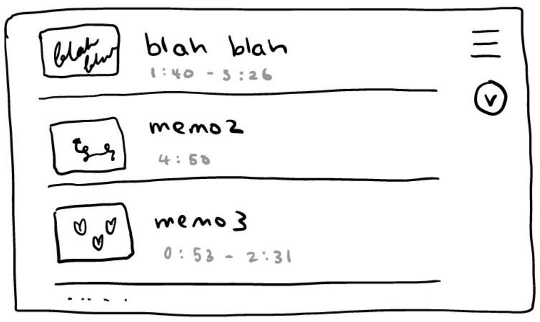

Figure 3: Bookmark View

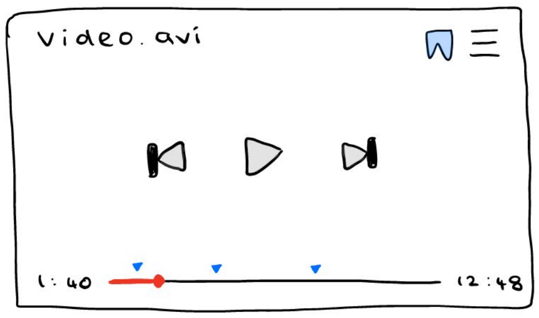

Figure 4: Bookmarks on the progress bar

The memos are saved as bookmarks when the contact points are lifted off. These bookmarks should be mapped to the playback position(s) of the video, where it was created (Figure 3). The type of file that is to be saved as memo should be different between the two modes. In Two-Finger mode, the memo will simply be saved as an image. On the other hand, one-finger mode should save the memo as an animated image or a video. Also the former requires one timestamp, when the latter requires two; the start and the end. The list of bookmarks can be seen accessed while playing the video as a sub menu. When selected, the memo is displayed or replayed on top of the video, with its playback position mapped to the timestamp(s). In the playing mode, these bookmarks are indicated above the playing progress bar (Figure 4). Lastly, memo can be imported or exported in different formats.

IMPLEMENTATION

MemoTube prototype app was implemented as an Android app for a tablet, with the android version of 4.4 Kitkat. It was written in Kotlin using Android Studio IDE. As additional modules, we used ExoPlayer [3] as the video player, and Android Draw [2] as a canvas view for the note-taking mode. We tested the app using a Samsung Galaxy Tab E (SM-T560) tablet. Our configuration requires touch input through screen. Hereafter we describe our choices for each design element.

Note-Taking Mode

Due to the lack of time, we implemented only the Two-Finger mode. To detect the contact points, a touch listener is set on the player view. When more than one pointer count is detected from the touch event, the note-taking mode is activated while the video player is paused. The draw is cleared when it was not empty and set visible with semi-transparency of white color (Figure 5). We save the current play state as a flag to decide on the play state after the note-taking mode ends. That is, video is resumed only when the video was already playing.
When the pointer count of touch event is less than two with the note-taking mode flag on, it is time to save the memo and change the mode to video player. The draw view is set to invisible, while the player view to visible. To check whether something has been drawn or not, the bitmap is compared to the empty bitmap. Only the non-empty ones are saved. Lastly, bookmark list view is updated to include the new memo.

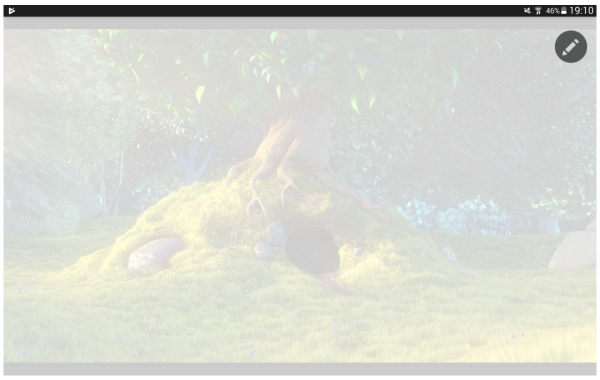
Figure 5 : Note-taking mode

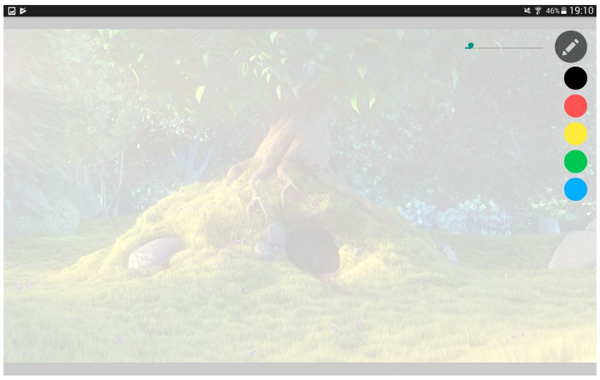
Figure 6: Editor Button

Pen Setting

During the implementation of the app, we thought that it would be useful for a user to change the color and the width of the pen while taking notes. For this reason, we added an editor button that is enabled (becomes visible) only when the note-taking mode is active. By pressing the editor button, users can change the color of the pen and also the width of the pen's line. To implement these functions, we used the DrawingActivity of Android Draw module. The button is placed at the top-right corner of the screen, as this is the position that doesn't occlude the notes that users take and that is easily accessible. When the editor button is pressed, the two settings become visible; the pen's colors (black, red, yellow, green, blue) below the button and the pen's width on the leftside of button (Figure 6). When the button is pressed again, the settings become invisible. The DrawingActivity has more settings like opacity, undo/redo, erase and save, some of which can be integrated in MemoTube in the future.

Bookmark Fragment
The bookmark list view implemented as a fragment. To add the fragment as a drawer, the main player layout was implemented as a DrawerLayout. So the bookmark view can be slide-open from the left(Figure 7). Also ListView was used to implement the bookmark list. The list basically loads the bitmap files that is stored in the folder in the external storage named 'Memotube' in the external storage, using the video title as a reference to file names. To sort the items by the mapped playback time, the list is cleared and recreated each time the new memo is added. However, this assumably makes the app slower or even crash under certain circumstances. When the item is clicked, the bitmap image of memo appears on top of the video, semi-transparently (Figure 8). The video playback position is changed and paused to the bookmark's mapped position. On clicking the exit button, the mode changes to video player.

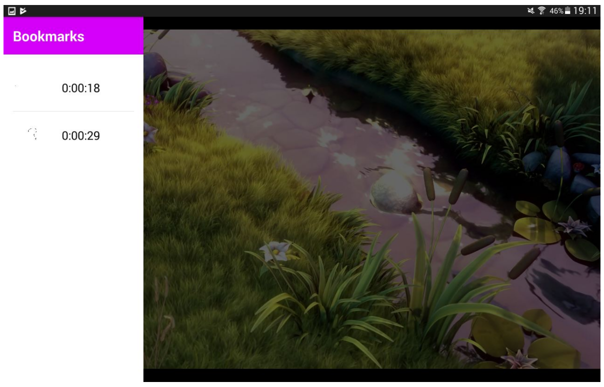

Figure 7: Bookmark List View

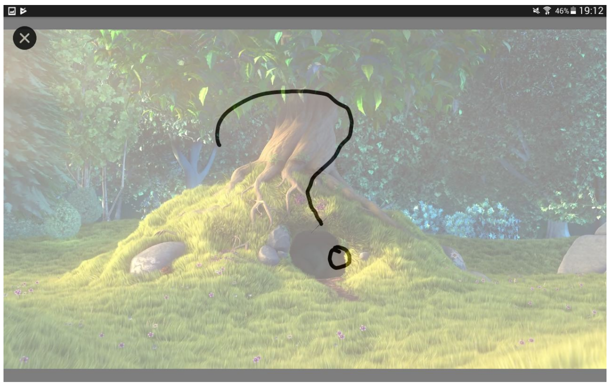
Figure 8: Displayed Memo

DIFFERENCES FROM THE PAPER
The differences between MemoTube and the paper[1] can be described in four aspects.

Data Type
The core difference between our implementation and the paper will be the type of media that is being dealt with. The main function of the app in our case is to play a video, whereas the paper deals with pdf documents.

Transience
The subject interaction of transience is different. In MemoTube, a note-taking input mode is activated when the additional contact point is placed, and any memo that is being put lasts until the contact point is lifted off. In the paper, what's being transient are the gestures. When the contact point disappears, the view is altered back to the bookmarked state of view before any gestures were done. So the interaction in our implementation should be called 'Transient Memo', instead of 'Transient Gestures.'

View Changing
In the paper, the view doesn't change when the additional contact point is placed. It only alters when the user actually performs the pan-and-zoom gesture. On the other hand, the view immediately changes in MemoTube when one finger is on, in a way that a semi-transparent canvas is layered on top of the video. The canvas disappears and the view changes back to normal video playing mode when the contact points are off.

Bookmark
The target of bookmark is also different. MemoTube saves the memo as a bitmap file, after the note-taking mode ends. The bookmarks are time-mapped onto the video, and can be displayed on top of the paused video at the saved playback time. It is not appropriate to say that the video playback position is bookmarked when two fingers are on, because the video is simply paused for the input mode. While in paper, the document state is bookmarked when the contact point is being placed, to be restored later when the point disappears.

ISSUES

There are several problems that we faced that still remains to be handled in our prototype app.

Slow Performance
First, its slow performance. There are specific moments where it gets noticeable. One example is when a user lifts off the additional contact point(s) to change from note-taking mode to video playing mode. The memo lingers a bit longer with its semi-transparency, while the video has already started playing underneath. Assumably, it takes time to save the bitmap file in the external storage. Another example is when the bookmark memo is loaded on top of the video after selected in the list by the user. Similarly, the loading of the bitmap file on top of the media player requires time. Because of this, the loading of the memo and the video's playback position change don't seem to be synced.

Out-of-Memory Error
We noticed that the app crashes when the video has more than certain amount of bookmark items. This specifically happens after the user tries to slide open the bookmark menu. Though the amount of items that triggers this error is not consistent, we observed that the app tends to crash with more than five bookmarks. We assume that this is due to the loading of bitmap thumbnails in the bookmark list, and also the frequent reconstruction of the list whenever new memo is added.

Unexpected Line Creation
Sometimes a line is drawn unintendedly when taking the memo. The line usually starts from the corner of the screen to the point where the user's note-taking finger is on. This is probably happening because of the screen has sensed the change of touches from somewhat minute movements of users' fingers. We couldn't figure out the way to fix it.

FUTURE WORK

Lacking physical time and skills, we couldn't manage to implement every element that are included in our concept. Along with what was addressed in the above 'Issues', the following four elements can be added to improved the usability.

One-Finger Mode

We realized that introducing one-finger mode is more complex than we thought. In one-finger mode, the video should be kept playing while taking a memo. This means that the bookmarked memo should be saved in format of a video, not an image. So the app needs to handle two types of data as bookmarks. Saving, accessing and replaying the memo videos on top of the video player will be a challenge in terms of memory and performance.

Bookmarks Indication on Progress Bar

Instead of plain progress bar in the video player, the bookmarks can be indicated above the bar. This was inspired by Youtube, where the positions of advertisement are shown as yellow point on the progress bar. This function would allow users to skim to the bookmarked playback position and find the target memo.

More Bookmark Options

It will be useful if there are more options to import and export the bookmarks in the bookmark list view. Importing means to map the external image source onto a specific playback position, and will allow user to view the image on top of that video position. Exporting might include options to share the file as different formats, such as png, jpeg, gif, pdf, or avi.

More Pen Settings

The pen setting could be more useful if it has more options like erasing the canvas of the note, undo/redo buttons and a color palette, where users can freely choose the wanted color. These options will give to the users more freedom on how they design their memo.

REFERENCES

1. Jeff Avery, Sylvain Malacria, Mathieu Nancel, Géry Casiez, and Edward Lank. 2018. Introducing Transient Gestures to Improve Pan and Zoom on Touch Surfaces. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 25, 8 pages. DOI: https://doi.org/10.1145/3173574.3173599
2. Divyanshu Bhargava. Android Draw. Retrieved October 5, 2018 from https://github.com/divyanshub024/AndroidDraw .
3. Google. ExoPlayer. Retrieved October 5, 2018 from https://google.github.io/ExoPlayer/ .