A Raycast Approach to Hybrid Touch / Motion Capture Virtual Reality User Experience

Ryan Spicer, Rhys Yahata, Mark Bolas, Evan Suma

Problem:

Our Virtual Environment (VE) uses a large touch screen device for user input (Figure 1). The head mounted display (HMD) and user's palms are tracked through an optical motion capture system. This data drives a puppeteered self-avatar. The motion capture system does not track the user's fingertips, and the physical touch screen is manually registered.

Since the user's fingertips are not tracked, several types of error can exist in the relationship between the user's physical hand and the self-avatar hand: Finger length and pose; hand tracker position; and touch screen registration.

These error sources can cause the user's fingertips and the self-avatar fingertips to not align. As a result, the point that the user appears to be touching on screen within the VE may not be the point his/her fingertip is actually touching on the physical touch screen (Figure 2). This creates a frustrating user experience.

Figure 1: The E2C2 / BlueShark system uses an optical motion capture system to track an HMD and the user's hands. A touch screen display provides additional input. The touch screen image is valuable to spectators observing the user in the HMD, and when the user is not wearing the HMD.

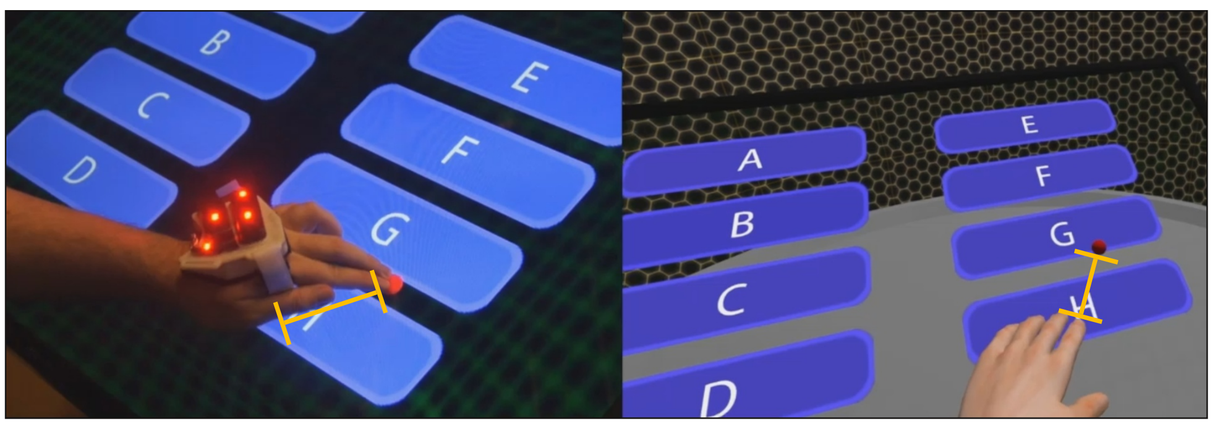

Figure 2: The error case. The touch point from the touch screen (red sphere) is accurate to the user's physical fingertip (left), but does not align with the self-avatar's fingertips in the VE (right). The incorrect button appears to be touched, in the HMD.

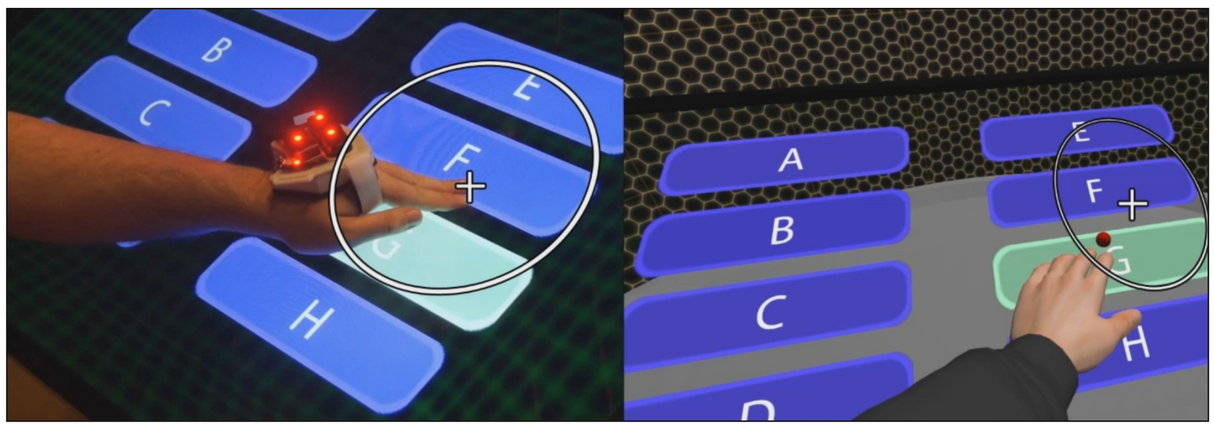

Figure 3: The corrected case. The physical touch point (white cross), is updated by raycasting through the self-avatar's fingertip in the VE (red sphere). The correct button is touched, from the user's perspective in the HMD.

Method:

We combine touch coordinate data from an infrared (IR) touch screen with hand position data from an optical motion capture system.

1) When the touch screen reports a touch, we project the touch into the VE's coordinate system and determine the closest self-avatar fingertip to the touch.

2) We check if the closest self-avatar fingertip is within a threshold radius of the projected touch point (Figure 3).

3) If the touch is within a threshold distance, raycast from the midpoint of the user's eyes, through the self-avatar fingertip, into the VE touch screen (Figure 4).

4) We forward the corrected touch event to the User Interface library for handling (Figure 3, right).

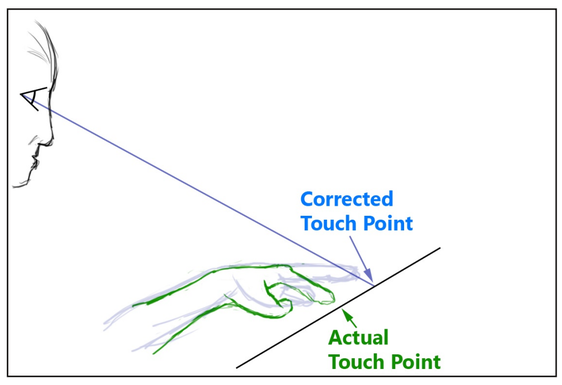

Figure 4: Side view of the raycast step. The user's physical hand is indicated in green, and the self-avatar hand is indicated in blue.

Future Work:

This approach may be refined and expanded. The current approach does not afford multi-touch gestures, as it assumes a single touch-point per hand. We could use knowledge about the tracked palm and human anatomy to draw inferrences about multiple touches in close proximity.

This approach can function even without a touch screen device. In demonstrations, we have used a blank sheet of glass as haptic feedback for a simulated touch surface. In this application, the Unity engine's collision handling is used to determine when the self-avatar fingertip touches the surface and generate a touch event.

Acknowledgements:

This work was supported by the Office of Naval Research through Award No. W911NF-04-D-0005-0041.

The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

USC Institute for Creative Technologies