TransparentHMD: Revealing the HMD User's Face to Bystanders

Christian Mai, Lukas Rambold, Mohamed Khamis
LMU Munich, Germany
Christian.Mai@ifi.lmu.de, Lukas.Rambold@campus.lmu.de,
Mohamed.Khamis@ifi.lmu.de

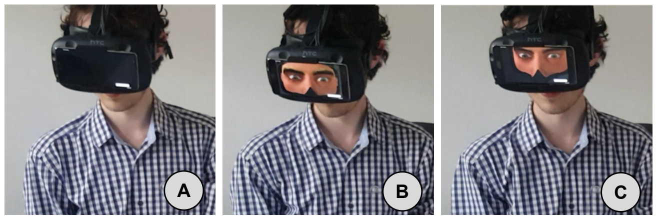

Figure 1: Our approach allows for bystanders to perceive the HMD user's face. (A) shows the situation as it is today. In (B), we show a 2D image of the user's face on a smartphone screen mounted on an HMD. While in (C), we show a 3D-face model that is projected according to the bystander's head position, which is detected using the front-facing camera of the smartphone (Perspective mismatch in (C) is created by the camera position and the tracked head position).

MUM 2017, November 26–29, 2017, Stuttgart, Germany
ACM 978-1-4503-5378-6/17/11.
https://doi.org/10.1145/3152832.3157813

Abstract

While the eyes are very important in human communication, once a user puts on a head mounted display (HMD), the face is obscured from the outside world's perspective. This leads to communication problems when bystanders approach or collaborate with an HMD user. We introduce TransparentHMD, which employs a head-coupled perspective technique to produce an illusion of a transparent HMD to bystanders. We created a self contained system, based on a mobile device mounted on the HMD with the screen facing bystanders. By tracking the relative position of the bystander using the smartphone's camera, we render an adapting perspective view in realtime that creates the illusion of a transparent HMD. By revealing the user's face to bystanders, our easy to implement system allows for opportunities to investigate a plethora of research questions particularly related to collaborative VR systems.

Author Keywords

Virtual Reality; Head-mounted Displays; Eyes; Face; Gaze

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

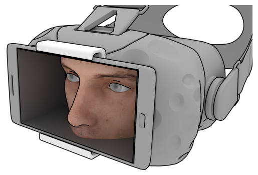
Figure 2: The aim of our system is to render the user's face on the front of the HMD in a way that would let bystanders perceive the HMD as a transparent one.

Introduction and Related Work

The recent fall in prices of head mounted displays (HMDs) encouraged a wider adoption of VR applications. However, while the new HMDs promise more immersive experiences particularly in VR applications, HMDs hide the user's face. The human eyes and face are strong tools for human communication. They are needed for example to regulate conversation flow, providing feedback, communicating emotional information, communicating the nature of interpersonal relationships and avoiding distraction by restricting visual input [6]. Additionally the face reflects emotions [2].

In this work, we report on our implementation of TransparentHMD. We 3D-printed a mount that holds a mobile device on the outward-facing surface of an HTC Vive. The mobile device shows the user's face in a way that gives an illusion of a transparent HMD. We showcase three versions. Figure 1A shows how the interaction is like today, with nothing presented on the HMD's outward-facing surface. Figure 1B shows the first version of our approach; a static 2D illustration of the user's face. While Figure 1C demonstrates the second version of our approach, which leverages a head-coupled perspective technique to allow the illustration to appear differently depending on the perspective of the bystander. More specifically, we track the head of the bystander through the front-facing camera of the mounted mobile device. This information is then used to render a 3D-face model projection in real time on the image plane, such that the observer has the impression of a transparent display (see Figure 2).

Previous work looked into how eye tracking can be used to animate the eyes presented on a front facing display [3] or even fully cover the user's head to hide the HMD with a view on the virtual scene [9]. In contrast, our work looks into the use of depth illusion in order to improve the impression of actually looking at the HMD user's face. This way, our approach has the potential to improve communication and interaction between HMD users and bystanders. There are many situations where such interaction can be useful. For example, recent work demonstrated the benefits of including bystanders in experiences of HMD users; Gugenheimer et al. found that this increases enjoyment, presence and social interaction [8]. Additionally, there is a plethora of work about collaborative environments in which non-HMD users collaborate with HMD users in. This can be in scientific context for example, VR training where the trainee wears an HMD while the trainer uses a PC [7, 13], building 3D scenes where the designer is a non-HMD user and an HMD user perceives the scene [4, 10], and collaborative virtual environments (CVEs) [5]. Or it can have a commercial background like presenting cars in an HMD based configuration tool [1]

Prototype Concept and Implementation

TransparentHMD at its core describes the idea of making HMDs invisible by adding a display to the outside. It renders an augmentation of the obstructed area of the user's face in a way as if there is no headset in the first place. For this to happen, the system actively tracks the position of the bystander to render a 3D-face model projection that is adjusted to the image plane according to the bystander's perspective. This way, the bystander has the impression of a coherent scene (see Figure 2).

Design Requirements & Considerations

This system is designed to foster communication between HMD and non-HMD users. As a result, the design of the proposed method has to meet not only purely functional requirements, but it should: (1) seamless integrate into the visual appearance of the hardware, (2) do not impose more face obstruction, (3) be affordable and portable, ideally without additional hardware for tracking the bystander's head for fast setup and cost effective implementation in various contexts, and (4) interdependency from VR-platforms and affiliated system for a future-proof implementation.

To achieve these goals, we decided to leverage a mobile device to display the user's face. We were able to integrate it seamlessly into the HMD by using a 3D-printed holder (requirement 1). The mobile device's size is comparable to that of the HMD hence it does not obscure the user's face further (requirement 2). The recent advancements in processing power and in visual computing allow performing head tracking directly on commodity mobile devices, hence our solution is portable, and does not need additional hardware (requirement 3). Finally, being a separate platform, our system can be connected to other hardware and software by, for example, integrating an eye tracker in the HMD, allowing for interaction with the virtual scene, or communicating between the mobile device and the software running the HMD (in our case, Unity 5) (requirement 4).

Furthermore, with double-sided smartphones becoming popular[1], HMDs that leverage mobile devices as displays such as Google cardboard and Samsung Gear VR can adopt our solution without any additional mobile devices.

Implementation
The general Android application architecture can be seen in Figure 3 and will be described in the following. For easier installation and to prevent the issues created by having two apps running at the same time on the android device – e.g. preventing apps in the background from accessing the camera and/or slow down or pause background apps to save battery – we decided to produce one single installable Android package. The components are a Unity 5.6.0f3 instance that includes the 3D model of the face and the Main Camera viewport that renders the perspective view presented on the screen. The relative X- and Y- positions of the bystanders face in the camera picture are updated every frame by the FaceTrackerLauncher. This also handles the activity of the FaceTrackerPlugin within our Custom Android Library, which is starting and handling permission for the camera and the variables given by the Mobile Vision API [2] included in the Google Play Services. To detect the position of the bystanders face, we used the center of the bounding box provided by the Mobile Vision API, which marks the center between the eyes with an update rate of 30Hz.

The 3D Model integrated in to Unity scene uses *PhilipFace03* and *PhilipEyes03* from the *Genesis2Male* model from DAZ3D[3], which is optimized for running on a smartphone and customized to match the impression of the users' face by using DAZ Studio Pro 4.9.3. We animated the face using "RandomEyes"[4] to generate random movements of the eyeballs, eyebrows, and eyelids. The random movements were focused around the bystanders direction in order to create the feeling of being looked at. The 3D model also includes the inner part of the HTC Vive [5]. Furthermore we are able to let the avatar look into certain directions on key press like downwards or upwards. The Unity scene also acts as a host for all other components on the smartphone, offering a basic setting UI switching between the modes shown in Figure 1, rendering out the model on the screen, and applying the head-coupled-perspective projection based on the tracking data from the front-facing camera that shows the bystander's position relative to the user. In

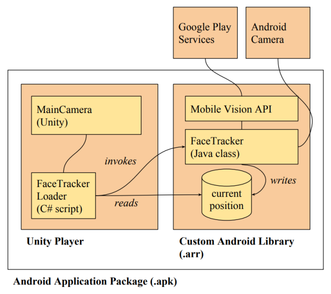

Figure 3: Our system architecture links different services with the a unity scene containing the face model. Everything is combined into one android package.

[1] http://abcn.ws/192Nc1v

[2] https://developers.google.com/vision/
[3] https://www.daz3d.com/
[4] http://crazyminnowstudio.com/unity-3d/lip-sync-salsa/
[5] https://sketchfab.com/models/4cee0970fe60444ead77d41fbb052a33

order to facilitate recreating our system we will describe our implementation of the Head-coupled Perspective (HCP) and the smoothing of the Data in the following.

Head-coupled Perspective (HCP)
An HCP can not be achieved by mere camera movements that are applied by the `FaceTrackerLauncher` script. The pure movement of the camera does not consider the changes in the virtual camera frustum created by the detected position of the bystanders head in relative position to the boundaries of the smartphones screen. When the face is observed from an angle for example, the display in the real world is not a rectangle but a trapeze, signaling that distances on the display, further away from the eye are squeezed, while distances closer to the eye appear longer. However without adaption, the virtual camera will still have a symmetrical view frustum. To achieve this perspective correction we adapt the virtual cameras frustum during vertex operation in the rendering pipeline of Unity.

Projection in Rendering Pipeline To create a mapping from 3D-coordinates to 2D-coordinates that can be displayed on a device, four steps are applied to the vertex: First, model transformations are applied to local model vertices, resulting in (x, y, z) world coordinates. Second, world coordinates are transformed to camera coordinates, where the camera is the origin of the coordinate system. As a next step a projection matrix and a normalization matrix are applied to transform all coordinates inside the frustum to normalized device coordinates (NDC). To display those on the target display, the normalized coordinates have to be mapped to the actual viewport with simple translation. The projection step is where changes in the perspective distortion can be made. The default projection matrix in OpenGL and therefore in Unity '`GL_PROJECTION`' is defined by the dimensions of the camera frustum as follows:

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Where n and f are the near and far culling planes and r the right and l the left, t the top and b the bottom are the offsets of the respective edge from the main camera ray at the height of the near clipping plane. The angles of the viewing volume and the depth of field are implicitly defined by the position n of the near clipping plane [12].

The C# script `ObliqueFrustum` applies the current projection matrix in the `Camera.projectionMatrix` attribute. For defining all variables in the base case of a frontal camera position, fixed height and width are retrieved from the dimensions of a proxy object that represents the opening of the HMD and the near clipping plane. The left bound on the near clipping plane is computed e.g. by $l = -\frac{witdth}{2}$, while the right clipping plane can be described by $r = \frac{witdth}{2}$.

To support camera movement caused by the face tracker, the frustum values and the resulting matrix are updated at every frame (60 fps). For the updated values of l and r we subtract from the the previously calculated fixed value the (horizontal) x-translation of the camera (`localPosition`). Variables t and b behave analogously with vertical translation. For distance to the near clipping plane n, TransparentHMD uses the distance to the proxy object and for the far clipping plane f a fixed large enough value to include every vertex in the scene. The result of different projection matrices can be observed in figure 1, where the nose is significantly distorted. When this is observed from the upper left corner the perspective distortion of the matrix would cancel out the perspective distortion of the real world, resulting in the illusion that the paper or screen has depth.

Smoothing If face tracking data images were directly applied to the perspective correction, quick jumps in values would result in unnatural flow of the face animation. Hence we smooth the face tracking data with a regulation technique, a p-Filter. It can be described as:

$$u(t) = K_p * e(t)$$

, where t is the time, measured in frames, $e(t)$ is the input signal (ie. the change in tracking data) $u(t)$ is regulating signal and thus the change, which will be applied to the render. $1 \leq K_p < 0$ denotes the amplification of the filter [11].

For $K_p = 1$ changes are directly applied to the outgoing signal. The smaller the value, the steadier the output signal becomes. However steadiness also results in a slower propagation of actual changes to the output. Hence a trade-off has to be made between responsiveness and stability. In this work we used a value of $K_p = 0.05$.

Technical Limitations

Our work comes with some limitations. First, the head-coupled projection in our implementation is restricted by the field of view of the smartphone's front-facing camera. However, advances in smartphone promise front-facing cameras with wider lenses, and future systems can employ a dedicated wide-lens camera (e.g., fisheye lens). Second, our system supports a single non-HMD user only. This means that if the user is surrounded by multiple bystanders, it would render the face according to one of them only. Future systems could choose which bystander to render the user's face to, depending on the bystander's gaze; if the bystander gazes at the user, then the system would adapt the animated face according to that bystander's position.

Conclusion and Future work

In this work we introduced TransparentHMD, a prototype that renders the face of HMD users to the bystanders. Rather than showing a static 2D picture of the eyes as in figure 1B, we instead render a 3D animation (Figure 1C) that reacts to the position of the bystander as detected by a front-facing camera of a smartphone mounted on the HMD. In future work, we intend to use eye tracking from within the HMD to animate the shown eyes accordingly. We also plan on building a complete pipeline that starts by a 3D scan of the user's face (e.g., using a Kinect) and then generating a 3D model of the face to use in TransparentHMD.

REFERENCES

1. AUDI AG. 2016. Audi Digital Illustrated - Audi VR experience. (2016). `https://audi-illustrated.com/en/CES-2016/Audi-VR-experience`

2. John Bassili. 1979. Emotion recognition: The role of facial movement and the relative importance of upper and lower areas of the face. *Journal of Personality and Social Psychology* 37, 11 (1979), 2049–2058. DOI: `http://dx.doi.org/10.1037/0022-3514.37.11.2049`

3. Liwei Chan and Kouta Minamizawa. 2017. FrontFace: Facilitating Communication Between HMD Users and Outsiders Using Front-facing-screen HMDs. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. ACM, New York, NY, USA, Article 22, 5 pages. DOI: `http://dx.doi.org/10.1145/3098279.3098548`

4. Karin Coninx, Frank Van Reeth, and Eddy Flerackers. 1997. A Hybrid 2D / 3D User Interface for Immersive Object Modeling. In *Proceedings of the 1997 Conference on Computer Graphics International (CGI '97)*. IEEE Computer Society, Washington, DC, USA. http://dl.acm.org/citation.cfm?id=792756.792856

5. Thierry Duval and Cedric Fleury. 2009. An Asymmetric 2D Pointer/3D Ray for 3D Interaction Within Collaborative Virtual Environments. In *Proceedings of the 14th International Conference on 3D Web Technology (Web3D '09)*. ACM, New York, NY, USA, 33–41. DOI: http://dx.doi.org/10.1145/1559764.1559769

6. Maia Garau, Mel Slater, Simon Bee, and Martina Angela Sasse. 2001. The impact of eye gaze on communication using humanoid avatars. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '01*. ACM Press, New York, New York, USA, 309–316. DOI: http://dx.doi.org/10.1145/365024.365121

7. Dominic Gorecky, Mohamed Khamis, and Katharina Mura. 2017. Introduction and establishment of virtual training in the factory of the future. *International Journal of Computer Integrated Manufacturing* 30, 1 (2017), 182–190. DOI: http://dx.doi.org/10.1080/0951192X.2015.1067918

8. Jan Gugenheimer, Evgeny Stemasov, Julian Frommel, and Enrico Rukzio. 2017a. ShareVR: Enabling Co-Located Experiences for Virtual Reality Between HMD and Non-HMD Users. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4021–4033. DOI: http://dx.doi.org/10.1145/3025453.3025683

9. Jan Gugenheimer, Evgeny Stemasov, Harpreet Sareen, and Enrico Rukzio. 2017b. FaceDisplay: Enabling Multi-User Interaction for Mobile Virtual Reality. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 369–372. DOI: http://dx.doi.org/10.1145/3027063.3052962

10. Roland Holm, Erwin Stauder, Roland Wagner, Markus Priglinger, and Jens Volkert. 2002. A combined immersive and desktop authoring tool for virtual environments. In *Proceedings IEEE Virtual Reality 2002*. 93–100. DOI: http://dx.doi.org/10.1109/VR.2002.996511

11. Jan Lunze. 2016. *Regelungstechnik 1*. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: http://dx.doi.org/10.1007/978-3-662-52678-1

12. Alfred Nischwitz, Max Fischer, Peter Haberäcker, and Gudrun Socher. 2007. Computergrafik und Bildverarbeitung. (2007), 608. DOI: http://dx.doi.org/10.1007/978-3-8348-9190-7

13. Jauvane Oliveira, Xiaojun Shen, and Nicolas Georganas. 2000. Collaborative Virtual Environment for Industrial Training and e-Commerce. In *IEEE VRTS*.