A universal launcher for glanceable AR headsets

Khushman Patel
College of Computing
Georgia Tech
Atlanta, USA
khushman@gatech.edu

Prof. Blair MacIntyre
College of Computing
Georgia Tech
Atlanta, USA
blair@cc.gatech.edu

***Abstract*—AR headsets have the potential to be an effective medium to display various information needed by users. In this project, we explore the paradigm of glancability for AR headsets by proposing a novel glanceable AR ”launcher” which uses the user environment to be unobtrusive while still being effective at providing information to the user. We implement a proof of concept of the proposed launcher, and showcase the interface through a Virtual Reality(VR) demo. We discuss the technical issues with a complete implementation and details aspects of the system that need further examination.**

***Index Terms*—Human-centered computing - Mixed/augmented reality; Human-centered computing - User interface design**

I. Introduction

Augmented Reality(AR) headsets are becoming more powerful by the day, and will soon be consumer technology. They are envisioned to be similar to computers, providing information according to our needs [1]. With the prevalence of information in the smartphone era, users have various information needs to be able to complete goals or take decisions, and these needs have been well studied [2] [3] [4]. To complete these information needs, multi-tasking will be essential for AR headsets, e.g. apps on smartphones. Providing a useful interface to access and interact with these apps, or feeds as we will call them for AR, is essential. Adaptive interfaces that mold themselves according to user behaviour and their environment have been envisioned before [5]. Various methods to unobtrusively provide information through glanceable AR have recently been studied for their suitability for different types of tasks. Lu, Feiyu et al. [6] state that the ”head-glance” and ”eye-glance” methods are preferred, with the the ”eye-glance” being preferred for ”continuous monitoring tasks”.

In this project, we explore the paradigm of glanceable AR through an adaptive interface for multiple feeds that can fulfill the user's information needs while being unobtrusive by fading to the environment when unneeded. We implement a Proof-of-concept system that explores such an interface, and envision an end-to-end system for a realtime multitasking launcher. The PoC interface can be accessed on a WebGL compatible device at http://khushmanpatel.com/projects/ar_launcher.html.

II. Background

Glanceable interfaces for different mediums have been explored, e.g. TapGlance [7]. TapGlance considers multiple levels of user attention while interacting it. Similarly, the peripheral awareness strategy in information design uses the user's peripheral attention to place information so that it is unobtrusive, while being easily accessible [8]. ”Periphery” can be defined as ”what we are attuned to without attending to explicitly” [9] [10]. Non-attentive monitoring of states or changes in simple secondary information can be effectively designed using this peripheral vision [11]. Embedding information in the periphery can potentially free users to focus on a primary objective while keeping track on less important tasks [12]. Systems designed for real-life tasks need to be flexible enough to accommodate all of the user's needs. AR is well suited for this task as AR headsets can track their user and their environment and can work well for such a peripheral interface [6].

Within the AR headset space, information access has been well studied. With information always available to the user in an AR headset, it needs to be presented such that it doesn't block the real world, while still being accessible to the user. Multiple studies have tried to focus on using the cognitive load on the user as a measure for the level of detail in the information presented to the user [13] [14]. Some even include environment information in displaying this information [15]. More complete AR systems to access information have been designed, e.g. ARWin, which offers a replacement to a traditional desktop workspace using virtual programs [16]. Virtual content positioning has been found to be important when considering a complete AR experience, as it cannot block important information in the real world [5] [6].

III. Motivation

AR technology is becoming increasingly feasible with AR libraries being integrated into smartphones, e.g. ARKit [17]. Consumer grade headsets are also becoming more powerful, e.g. the Hololens [18]. With this new computing medium becoming consumer-ready, application development will increase for this platform. Promoting and keeping open standards relevant is important to maintain inter-operability between different applications on the platform as this happens.

To that end, the ARML 2.0 standard, published in 2015 [19] has not seen major updates to its core specifications. This paper aims to test out the applicability of ARML 2.0 using the latest updates in AR libraries, especially WebXR as of April 2020, to create a ”laucher” for a glancable AR headset. Built on ARML 2.0 feeds, it aims to provide the user a realtime dynamic interface to their native applications. The launcher

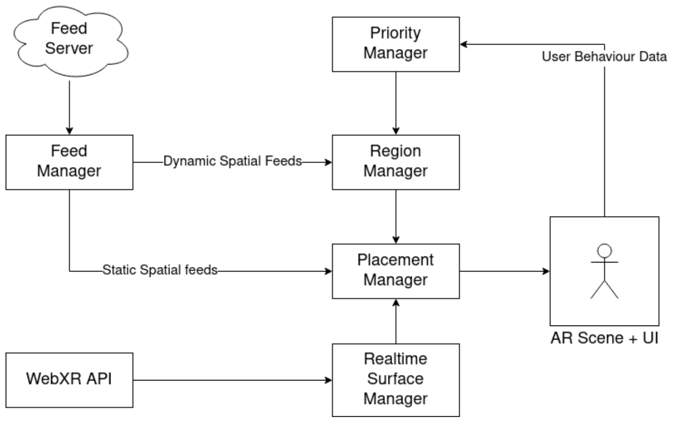


Fig. 1: The proposed end-to-end launcher system design

explores the use case of a dynamic glanceable interface that can unobtrusively provide the user with the needed information using the environment effectively.

Through the implementation of the launcher, we intend to glean insights into what aspects of the ARML standard might need extensions to keep them relevant with the evolving use cases of AR. We discuss these extensions later on in the paper.

IV. Implementation

A. Proposed system

The proposed system includes and end-to-end solution to implementing a realtime launcher on AR headsets. This incorporates a tie-in with the WebXR API [20] to analyze the environment in real time, learning through user behaviour, and feed servers that communicate with the headset and maintain a relevant stream of data to the user. A figure depicting the parts of the system can be seen in Fig 1. Each of the constituent parts and their functions are described below.

1) Realtime Surface Manager: The realtime surface manager continually streams geometry information about the user's environment from the WebXR API and keeps track of previously tagged or new surfaces which are valid to display feed information to the user on.

2) Feed Manager & Server: The feed manager communicates with the feed servers to ensure that it has any feed updates necessary. The feed server also serves temporal feeds, e.g. a new message notification, and geo-spatially relevant content, e.g. restaurant listings in the user's viewport. The feed manager contains an ARML parser that parses any updates in the feeds and informs the Region or Placement Manager of any changes.

3) Region Manager: A region is a collection of feeds which is displayed as a unit to the user. Regions are dynamically positioned around the user, and they can be pinned to a surface or follow a user around. A region can contain a single feed or a collection of feeds organized together. The Region Manager keeps track of these regions and handles management commands with regard to regions.

4) Priority Manager: The priority manager manages the relative priority of regions between each other. This node aims to dynamically provide the user with the most relevant feeds according to their usage. It learns the priorities of apps as the user uses the system and provides more information about their habits to the system. Over time, the priority manager aims to keep the most relevant feeds at the best position with respect to the user.

5) Placement Manager: The organizational core for all the feeds that are subscribed into the system. The placement manager arranges dynamic regions according to the environment, manages spatially static feeds, and handles temporally relevant updates, e.g. notifications.

6) Glanceable User Interface design: The primary aim of the launcher is to display information from subscribed feeds to the user. The user interface needs to provide feed and region management commands, and can be minimal. The necessary UI functions needed are:

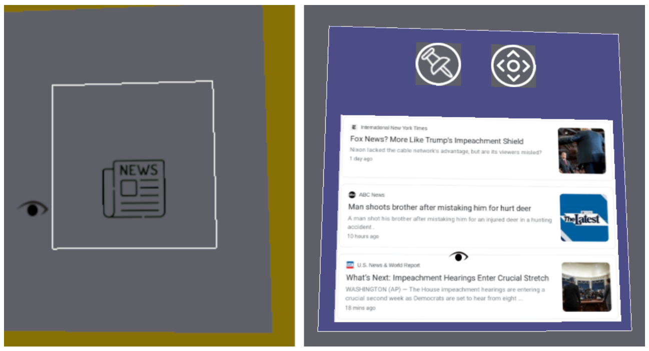
- Subscribing and unsubscribing to a new feed
- Enabling/Disabling feeds
- Grouping feeds together
- Pinning a region to a surface

Glanceable is said to be the quality of letting users get information quickly and with low effort [21]. Glanceable visuals are fastest interacted with when they have a high degree of symbolism, or are text [22]. To that end, all feeds described in this interface have a primary and secondary context. The primary context provides a visual to distinguish between feeds when quickly navigating to a particular feed. The secondary context is shown when the user is looking at a feed, and adds more detail to the feed and the user interface. An example of both can be seen in Fig 2.

Apart from the contexts, we make sure that the UI is highly responsive to where the user glances to it. Every element gives feedback for a glance to the user. We define a "Glance Attention Area"(GAA) as the relevant focus area around the tracked eye of the user. GAAs are typically circular to denote the area the eyes are looking at, but can be different geometries depending on the application. To make the interface appealing, the primary context is switched to the secondary whenever the context intersects with the user's GAA. The actual area of the GAA depends on the application and how detailed its interface needs to be. In our case, the GAA is wide when the user is looking at primary contexts to focus on, and narrows down when the user is interacting with the secondary context, e.g. to click on an icon in Fig 2.

B. Implemented Proof Of Concept

The implemented PoC provides the user with an experience similar to the interface of the proposed system in a WebVR environment. We provide the user with a house environment to simulate them moving around, where they interact with the interface described in this paper. The constituent parts of the PoC can be seen in Fig 3. The implemented modules are described below.



International New York Times
Fox News? More Like Trump's Impeachment Shield
Nixon lacked the cable network's advantage, but are its viewers misled?
1 day ago

ABC News
Man shoots brother after mistaking him for hurt deer
A man shot his brother after mistaking him for an injured deer in a hunting accident .
10 hours ago

U.S. News & World Report
What's Next: Impeachment Hearings Enter Crucial Stretch
WASHINGTON (AP) — The House impeachment hearings are entering a crucial second week as Democrats are set to hear from eight ...
18 mins ago

Fig. 2: Primary Context(left, highly symbolic) vs Secondary Context

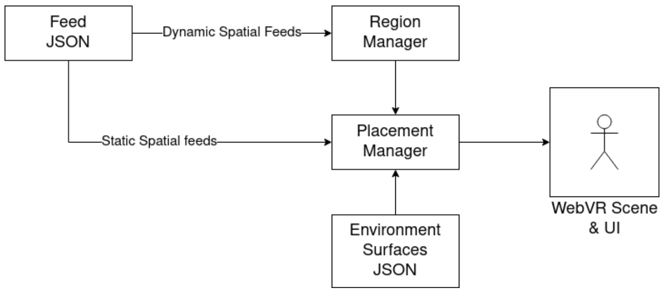


Fig. 3: Implemented PoC system design

- **Region Manager** - The Region manager manages the dynamically positioned feeds. It keeps track of enabled/disabled feeds and groupings of different feeds.
- **Placement Manager** - The placement manager arranges dynamic regions in relevant surfaces around the user and places spatially static feeds in the scene.
- **JSON files** - The subscribed feeds and environment surfaces are currently hardcoded in a JSON file and provided as configuration options to the PoC.

1) Glanceable User Interface design: The implemented UI interface functions are:

- Enabling/Disabling feeds, in Fig 4
- Grouping feeds together, in Fig 5
- Pinning a region to a surface, in Fig 6

The glanceable concepts discussed in the proposed system have been implemented in the PoC. A live version of the PoC can be opened on any WebGL compatible device here (http://khushmanpatel.com/projects/ar_launcher.html).

V. Discussion

A. Content

We will need to define the types of content which will be displayed through the primary and secondary context of the feeds, and the parameters for how it is displayed.

Content can be organized as in Fig 7. Content here refers to the actual data being displayed, a VisualAsset in the ARML 2.0 spec. All content used in the PoC has a primary and secondary context to display. While there could be content without a secondary context, it would either be static if it is highly symbolic, or in case of more detail, not be suited to be perceived in a glance. Thus the addition of contexts to feed data would be highly beneficial for a glanceable experience.

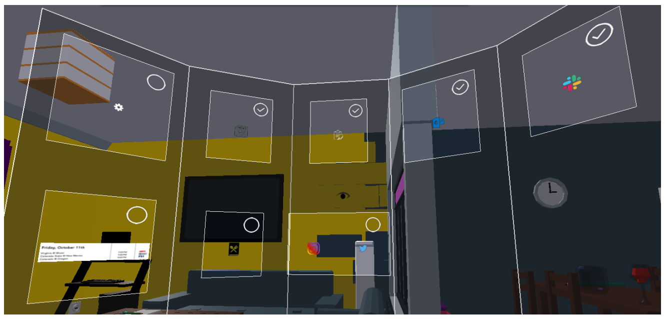
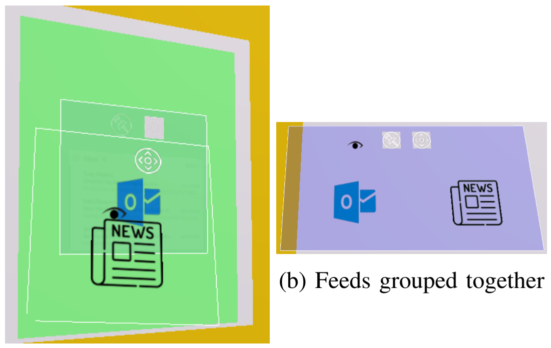


Fig. 4: Menu to enable/disable feeds



(a) Dropping one feed on another

(b) Feeds grouped together

Fig. 5: Grouping two feeds together

B. ARML 2.0 Extensions

1) Spatial Details: Spatial details are metadata information needed to define where to position a feed. They can be geo-static, which is a latitude/longitude if data is sent by a feed server, or a local surface that the user has pinned a feed to. This is well defined in the ARML 2.0 spec, and covers all use cases.

Follow the user feeds are dynamically positioned. The launcher positions them in realtime by parsing usable surfaces around the user. Priority between feeds is learned through user behaviour, and no further data is needed from a feed server to position these.

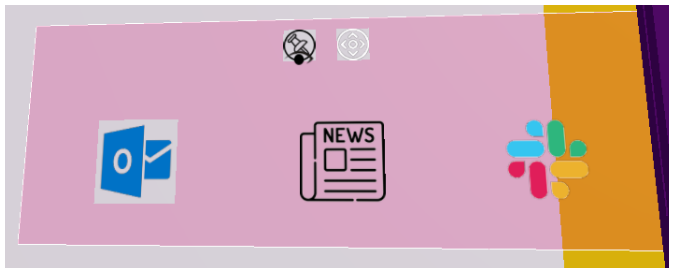


Fig. 6: A pinned region

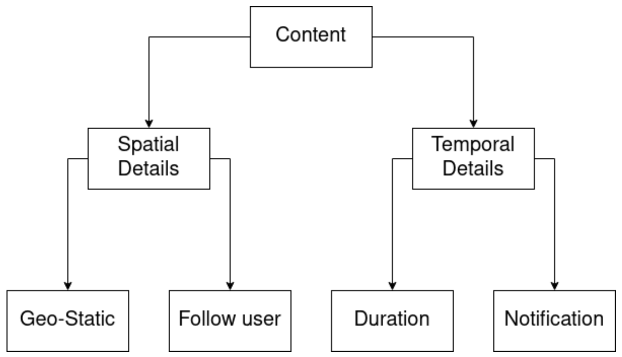


Fig. 7: Content organization

The ARML 2.0 spec defines an ARAnchor as an anchor relative to the real world. It has three types:

- **Geometry** Either a Point, LineString or Polygon, described with spatial coordinate tuples.
- **Trackable** A visual pattern that is detected in the camera stream.
- **RelativeTo** An Anchor relative to other objects (e.g. another ARAnchor); useful to create large scenes relative to another Trackable.

A further type, "Follow", which defines content which will stay with the user as they move will complete the toolset supported by the specification for further modern use cases.

2) Temporal Details: Temporal details are metadata information needed to defined when to show a feed. This can be a particular duration, e.g.

- **inf** This defines a feed which permanently stays with the feed manager till the user explicitly unsubscribes to it.
- **Time Duration** e.g. recurring 8AM-11AM show exercise feed, or Mar 28-Mar 30 2020 show event livestream.

It can also be a notification: a push notification similar to those used in smartphones, pushed from a feed to user e.g. a new message on a messaging application.

The ARML 2.0 spec defines a Condition interface which defines the constraints to show a VisualAsset. The current conditions possible are:

- **DistanceCondition** DistanceCondition allows VisualAssets to be activated and deactivated based on the distance of the user to the anchor.
- **SelectedCondition** The selected condition allows VisualAssets to be activated and deactivated based on the selected-status of the Feature or Anchor associated to it.

Further expansion of this interface can easily accommodate the aforementioned temporal details necessary to specify when to display a VisualAsset.

VI. Future Work

The implemented PoC shows promise for the creation of an end-to-end AR launcher. While most UI features have been implemented, there is scope for addition of multiple modules from the proposed system. The implemented PoC has statically assigned feeds and surfaces in the WebVR environment. The region manager deals with enabled user feeds, but feedback from actual user usage is not accounted for, and can make the system more functional in terms of displaying relevant content to the user. Dynamic regions are well implemented in the PoC, but geo-static feeds don't have the same amount of interaction and responsiveness. Further work in developing an interface for the same is needed. A feed manager which can manage subscriptions, handle feed content dynamically and parse updates from the server will add a functional backend which will help the user utilize the system for their use case.

Integration with the WebXR API is possible with major browsers and libraries supporting the API. Though the API is not mature enough for the proposed system yet, with the advent of AR headsets which can sense the environment around them, the system will be able to parse surfaces in the environment around the user in real time and display the relevant content accordingly. Implementation of the realtime surface manager will make the experience a "truer" form of AR.

Further work studying the effectiveness of information access and management using this system needs to be done. A user study focusing on the unobtrusiveness of the the interface to ensure glanceability is important. The effectiveness of the system in helping users learn and retain information is another important factor to be considered. Minimalism and the number of options provided to the user is another aspect of the system to look at, to ascertain if the system has achieved the desired level of simplicity and customizability.

Acknowledgment

Prof. MacIntyre for being supportive throughout and always asking the right questions. The good folks at Google for sponsoring a Research Assistantship during Fall '19.

References

[1] S. K. Feiner, "Augmented reality: A new way of seeing," *Scientific American*, vol. 286, no. 4, pp. 48–55, 2002.

[2] D. Dearman, M. Kellar, and K. N. Truong, "An examination of daily information needs and sharing opportunities," in *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, 2008, pp. 679–688.

[3] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan, "A diary study of mobile information needs," in *Proceedings of the sigchi conference on human factors in computing systems*, 2008, pp. 433–442.

[4] K. Church, M. Cherubini, and N. Oliver, "A large-scale study of daily information needs captured in situ," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 21, no. 2, pp. 1–46, 2014.

[5] W. Lages and D. Bowman, "Adjustable adaptation for spatial augmented reality workspaces," in *Symposium on Spatial User Interaction*, 2019, pp. 1–2.

[6] F. Lu, S. Davari, L. Lisle, Y. Li, and D. A. Bowman, "Glanceable ar: Evaluating information access methods for head-worn augmented reality."

[7] D. C. Robbins, B. Lee, and R. Fernandez, "Tapglance: designing a unified smartphone interface," in *Proceedings of the 7th ACM conference on Designing interactive systems*, 2008, pp. 386–394.

[8] J. Cadiz, G. Venolia, G. Jancke, and A. Gupta, "Sideshow: Providing peripheral awareness of important information," 2001.

[9] M. Weiser and J. S. Brown, "The coming age of calm technology," in *Beyond calculation*. Springer, 1997, pp. 75–85.

[10] ——, "Designing calm technology," *PowerGrid Journal*, vol. 1, no. 1, pp. 75–85, 1996.

[11] M. R. Endsley, "Design and evaluation for situation awareness enhancement," in *Proceedings of the Human Factors Society annual meeting*, vol. 32, no. 2. SAGE Publications Sage CA: Los Angeles, CA, 1988, pp. 97–101.

[12] P. P. Maglio and C. S. Campbell, "Tradeoffs in displaying peripheral information," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2000, pp. 241–248.

[13] B. Rhodes, "Wimp interface considered fatal," in *IEEE VRAIS'98: Workshop on Interfaces for Wearable Computers*, 1998.

[14] M. Billinghurst and T. Starner, "Wearable devices: new ways to manage information," *Computer*, vol. 32, no. 1, pp. 57–64, 1999.

[15] D. Lindlbauer, A. M. Feit, and O. Hilliges, "Context-aware online adaptation of mixed reality interfaces," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, 2019, pp. 147–160.

[16] S. Di Verdi, D. Nurmi, and T. Hollerer, "Arwin-a desktop augmented reality window manager," in *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* IEEE, 2003, pp. 298–299.

[17] Apple arkit homepage. [Online]. Available: https://developer.apple.com/augmented-reality/arkit/

[18] Microsoft hololens. [Online]. Available: https://www.microsoft.com/en-us/hololens

[19] M. Lechner, "Ogc augmented reality markup language 2.0 (arml 2.0), version 1.0." 2015.

[20] The webxr specification. [Online]. Available: https://github.com/immersive-web/webxr

[21] T. Matthews, "Designing and evaluating glanceable peripheral displays," in *Proceedings of the 6th conference on Designing Interactive systems*, 2006, pp. 343–345.

[22] T. Matthews, D. Blais, A. Shick, J. Mankoff, J. Forlizzi, S. Rohrbach, and R. Klatzky, "Evaluating glanceable visuals for multitasking," Technical Report EECS-2006-173. UC Berkeley, Tech. Rep., 2006.