3D App Launcher

MIŁOSZ GŁOWACZEWSKI, ROLAND KRISZTANDL, and TAMÁS CSIZMADIA

Currently, most of the virtual and augmented reality (VR/AR) launchers are adaptations of 2D screen-based launchers, such as the Windows start menu or phone interfaces. We propose an alternative solution where app icons are located on the surface of a sphere that can either be around the user or be placed in front of them. Two interaction types, poking and pinching, were developed and compared in terms of their usability and intuitiveness. To evaluate the effectiveness of the proposed solution, a user study was conducted with fellow students who provided feedback on its performance and usability. During the study, it was discovered that the poking interaction was more intuitive but significantly more challenging to implement effectively.

1 INTRODUCTION

Augmented/Virtual reality (AR/VR) has become an increasingly popular medium for both entertainment and professional applications, offering immersive experiences that engage users in novel and intuitive ways. A crucial aspect of the AR experience is the ability to efficiently and effectively navigate and interact with the virtual environment. One such component is the app launcher, which serves as a central hub for users to access various applications and services within the headset. Despite its importance, the Oculus Quest 2 still uses an app launcher that is based on 2D touch interfaces. This however does not fully benefit from the possibilities of the headset. It does not make use of available 3D space and hand gesture interactions. Appropriate design choices can significantly impact user experience, efficiency, and satisfaction. Our motivation is to improve the usability of app launchers, enhancing the overall AR experience for users across various applications. Using the headsets' full possibilities could potentially increase the intuitiveness and speed of interactions. [3]

In our prototype, we implemented both app launcher types and interaction methods, allowing users to rotate the sphere along a horizontal axis, and either poke or pinch to select and launch applications. The end result showcases the differences in user experiences based on the chosen app launcher type and interaction method.

Our study is centered around the following goals: exploring the usability of different app launcher types and interaction methods in Augmented Reality and identifying the most effective solutions for enhancing user experience.

2 BACKGROUND

As a starting point, we tested some already existing AR/VR app launchers, to experience their pros and cons.

On Microsoft Hololens 2's start menu, the user can see general settings such as network information, volume settings, etc, and up to 9 pinned app icons on a 3 by 3 grid. If there are more, up and down buttons will appear to go to the next/previous page. On the side, we can press a button to see the list of all apps. After pressing this button a 2 by 5 grid will appear which contains the first 10 apps. Similarly to the pinned apps menu, the user can go to the next/previous page. Both the start menu and the app menu is a relatively small 2D plane located close to the camera, which makes it easy to use with the poke interaction, which is the default interaction technique on the Hololens 2.

The other device we tested was the Meta Quest 2. Its app launcher has 4 columns and as many rows as needed to show all the apps, since the user is able to scroll down on the list. The icons are relatively big, therefore it's easy to launch any desired app using the ray interaction technique,

Authors' address: Miłosz Głowaczewski, 202202860@post.au.dk; Roland Krisztandl, 202202868@post.au.dk; Tamás Csizmadia, au724747@uni.au.dk.

which is the default interaction on the Quest 2. The launcher is a curved 2D plane, approximately 1-2 meters from the user.

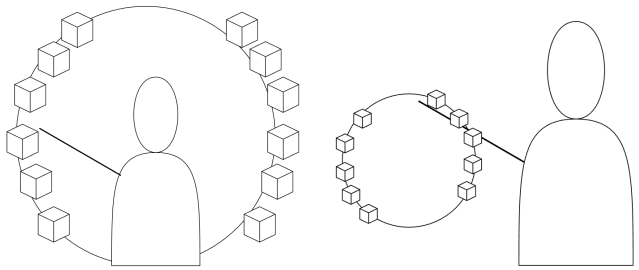
We noticed that both of these launchers were adapted from 2D screen-based launchers, like Windows start menu and phone launcher. Although this approach works well in practice, they don't utilize all possibilities of the headsets, especially the whole available 3D space. Both headsets use different interaction techniques, which shows that there are still no standardized paradigms for the interactions. [4]

3 CONCEPTS

Our main idea is that we place the app icons on the surface of a sphere, instead of a simple 2D plane, as seen in the already existing AR/VR app launchers. We propose two different states (or views) that can be more useful in different scenarios.

In the central view (Figure 1a) the user would be standing in the middle of the sphere, which provides a more immersive experience.

In the frontal view (Figure 1b) the launcher would appear in front of the user. This state was specifically developed as a futuristic concept when the users are not constrained to a room but are able to explore the world freely. In an open environment (for example, walking on the street) the central view should not be used, as it would limit the spatial awareness and visibility of the user.



(a) Concept of Central view. (b) Concept of Frontal view.

Fig. 1. Concept arts for the different designs.

We developed two different groups of interaction techniques to select an app and to rotate the launcher on the horizontal axis. The poking-based interaction requires direct contact with the app icon or the sphere itself for rotation. Meanwhile, the pinching-based interaction allows users to select applications from a distance, based on a ray cast from their eyes through their fingers. Both methods offer different levels of user engagement and control.

The potential pros and cons of each view and interaction method include:

- Central view: increased immersion, accuracy, and spatial awareness within the launcher.
- Frontal view: greater awareness of surroundings, but less immersive and less spatial awareness within the launcher interface.
- Poking interaction: direct and intuitive, but requires precise hand-eye coordination and may lead to accidental selections. [1]
- Pinching interaction: allows for selection from a distance, potentially more accurate, but may be perceived as less intuitive, as it is not natural for humans. [2]

The primary research questions addressed in this study encompass:

- Is utilizing a spherical app launcher in an augmented reality environment both effective and intuitive for users?

- How do the poking and pinching interaction methods compare in terms of workload?
- How do the poking and pinching interaction methods compare in terms of user satisfaction?

4 IMPLEMENTATION

We created a prototype app for the Meta (Oculus) Quest 2 device using the Unity Game Engine 2021.3.21f1 and the Oculus Integration SDK. Additionally, we also used Android Logcat for debugging.

During our work, we didn't use any special packages for the different interaction techniques (except for pinch detection), instead, we implemented everything ourselves. It required substantially more amount of work but gave us full control over the end results. Alternatively, we could have utilized the poke interaction from Oculus Interaction SDK[1] or from the MRTK the pressable button prefab[2] and the object collection organiser[3]. However there is a possibility that some limitations would appear along the way.

Our implementation is available on github: krr0land/ARAppLauncher

4.1 The launcher

As the first step, we added the `OVR Camera Rig` gameobject, configured the passthrough, enabled the hand-tracking and added the hand mesh prefabs. Additionally, we enabled the Physic Capsules on the hands, so that we could detect object-hand collisions.

We created two prefabs, one for the launcher itself and one for the apps.

The App prefab (simply called `Cube`) is a simple 3D cube, with two scripts attached to it (`DetectCollision`, `DetectPinch`), which are used by the interaction methods. This object has a child object, which is an enlarged semi-transparent cube, which is used to visualize an initial selection of the app for the poking interaction.

The AppLauncher prefab is a semi-transparent sphere, with one script attached to it (`Sphere Arranger`), which distributes a list of gameobjects equally on the surface of the sphere using a constrained version of the Fibonacci Sphere algorithm. This prefab has a child object, which is another semi-transparent sphere, but the triangles of the rendered mesh are inverted so that it can be visible from the inside.

We created a new child object for the CenterEyeCamera called `Launcher`, which has several scripts attached to it. One of these scripts is the `StopRotation`, which allows the object to move with its parent (aka the camera), but not rotate with it.

Another script attached to the `Launcher` is the `SpawnLauncher`. The `awake()` function instantiates the `AppLauncher prefab` and all the apps. The textures on the apps are loaded from file using the `AppCatalog` class. This script also detects the opening and closing of the application and provides a function to change the state of the launcher (frontal or central). When an app is selected by the different interaction methods, the `SelectApp()` function will be called.

The last two scripts attached to the `Launcher` are the `RotateLauncher` and the `SelectInLauncher` script, which implements the rotation and selection interactions. Both scripts have an interaction type enum, which allows us to change the used technique. (Since they are not dependent on each other, we could also combine the different interactions, but due to the inconsistency created by the mixed techniques, we didn't explore these options.) The rotation itself is independent of the

[1] https://developer.oculus.com/documentation/unity/unity-isdk-create-poke-interactions/

[2] https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/button?view=mrtkunity-2022-05

[3] https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/object-collection?view=mrtkunity-2022-05

interaction type since it only requires the current and previous hand position which is then used to calculate the angle between the two positions. The rotation is applied on the horizontal axes.

4.2 Pinching interaction

The pinching interaction allows the user to select an app or rotate the sphere using a pinching gesture. To initiate this interaction, the user brings their index finger and thumb together. To end it, they have to release the pinch.

To launch an app the user simply pinches and releases without any significant movement in a short amount of time. The location of the pinch should be between their head and the icon they want to select, essentially creating a virtual ray from their head towards the pinch point. The first icon the ray intersects with gets selected.

For rotating the sphere, the user maintains the pinch and moves their fingers along the desired rotation direction. The system continuously tracks the movement of the pinched fingers and rotates the sphere accordingly. The interaction ends when the user releases the pinch.

4.3 Poking interaction

The technique requires the user to poke an icon to select an app and poke the surface of the sphere and move the finger around to rotate the sphere. To end the interaction, the user must take their finger out of the surface.

The icons are located on the surface of the sphere, thus making the beginning of both interaction techniques similar. To differentiate between app selection and rotation, we drew inspiration from classic 2D mouse interaction. Specifically, if the user moves their finger by a pre-defined threshold while it goes through the sphere surface, the gesture is interpreted as a rotation. Conversely, if the finger is taken out from the surface without much movement, it is interpreted as an app selection.

The poking interaction starts when a collision between the index finger and the sphere outline or icon is detected. From this moment the initial position of the finger is saved and the application starts to keep track of the distance of the finger from the initial position. The interaction ends when the user takes the finger out of the sphere. Based on the movement of the finger the interaction is interpreted as app selection or rotation.

4.4 Tasks in the app

For the evaluation, we implemented a simple task system in the prototype application. When the participants were given the headset, they could see a text in front of them, with the next task that they are supposed to complete. The task described the application to launch, the interaction technique to use, and the state of the launcher (frontal or central). When the desired app was successfully launched, the next task was shown. Using the `A` and `B` buttons on the controller, we were able to jump to the next task or switch to the other task groups.

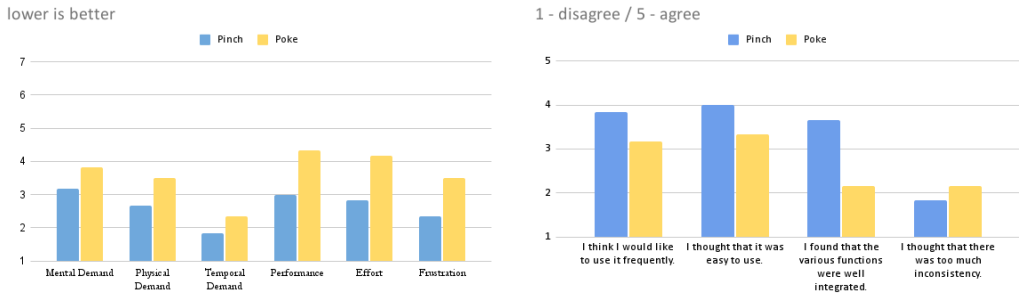
5 EVALUATION

In our research, participants were invited to test and evaluate our demo. We introduced them to the primary interactions and the two states of the sphere before asking them to perform tasks. There were two task routes: one focused on testing the pinch interaction, and the other on testing the poking interaction. Initially, participants engaged with the central sphere, followed by the frontal one. Due to concerns about the number of participants, we did not prioritize counterbalancing the two spheres, but we did for the two interaction methods. After each interaction type, participants were asked to complete a questionnaire.

For each interaction type, we designed four tasks, with two tasks per view. In a task, users needed to open the app launcher, initiate an app, and if successful, the instructions for the subsequent task would appear.

Overall, users were able to complete all tasks, with only a few instances where the launched app did not correspond to the instructions. However, everyone managed to initiate the correct app after several attempts.

We noted that the pinch interaction was faster, more accurate, and generally preferred by users. This observation is supported by the responses to our questionnaire Figure 2, which combined elements of the NASA TLX and System Usability Scale (SUS). The results indicate that pinching was less mentally, physically, and temporally demanding, as well as more effortless, accurate, and efficient. Users favored pinching, while they found poking to be less well-integrated.



(a) NASA TLX

(b) System Usability Scale (SUS)

Fig. 2. Responses to our questions

Participants also shared their personal impressions of the interactions:

- Pinching for rotation was highly intuitive
- Pinching for selection was more difficult due to the lack of feedback on what the user was about to select
- The large sphere was easy to rotate, either by swiping or through physical movement
- The small sphere provided a better overview of the available apps
- The poking interaction felt more intuitive
- Allowing rotation on all axes could be beneficial
- Navigating the small sphere with poking was more challenging due to frequent accidental selections
- Activating selection on the large sphere was difficult since icons were large, and poking required users to enter and exit at the same location

6 DISCUSSION

6.1 Is utilizing a spherical app launcher in an augmented reality environment both effective and intuitive for users?

The majority of the participants answered that they would like to use the launcher frequently and that it was easy to use.

When asked about the features, they liked in the prototype, the participants focused on the spheres. They commented that they liked the amount of icons, that a sphere can present. They also

complimented the ease of searching for the icon. On the other hand, there were comments, that the rotation should be possible in all axis, instead of horizontal only.

6.2 How do the poking and pinching interaction methods compare in terms of workload?

The NASA-TLX and Measure of Usability questions all received better feedback for pinching interaction. We think, that one of the reasons for that is the higher implementation quality for pinching interaction. For both interactions, we based the implementation on 2D touchscreen interactions. This worked for pinching because casting a ray can be interpreted as a 2D projection of the visible space by the user. This way the ray can be moved only in two axes. The poke interaction however is happening in 3D space. While it is easy to use for touch screens, it is not as easy for AR headsets. One main problem is higher physical fatigue as there is no surface to counter the poke movement. Another problem is the differentiation of rotation and poke interaction. Rotation is detected when the finger moves by a specified threshold when colliding with the sphere. However, poking gestures with limited movement on the other axis is not as easy in 3D space.

Another observation is that pinching requires less hand movement. Which makes it faster and less tiring to use. Some participants commented that selecting an app with poking was troublesome because it had to be very precise so that it wouldn't be interpreted as a rotation.

6.3 How do the poking and pinching interaction methods compare in terms of user satisfaction?

Pinching did not only require less workload but it was generally thought of as a better interaction in some cases. One of the participants observed that while poking was more intuitive for app selection, the rotation was more intuitive with pinching.

One participant commented that it was not intuitive that it is not a pinch itself that opens the app, but the ray going through the pinch.

The feedback from participants was mostly expected. While the poke gesture, in theory, is more intuitive because of the smartphones that we use every day, it doesn't work that well with AR headsets. Pinching might be generally better to use, however, requires getting used to that.

7 CONCLUSION AND FUTURE WORK

In this study, we presented a sphere-based app launcher for AR/VR purposes, that utilized the 3D environment more than the current launchers found on this platform. We also experimented with two different interaction techniques, that can be used to interact with the launcher itself. We developed a demo for our concepts so that we could compare the interaction types. We conducted a user study that revealed that the poking interaction was more intuitive but also more challenging to implement effectively compared to the pinching interaction. The evaluation of the prototype demonstrated that participants were able to complete tasks successfully using both interaction methods. However, the pinching interaction was favored by users due to its faster and more accurate nature. The participants overall liked the spherical layout and said that it can present a lot of icons.

Future research is needed to address the implementation challenges associated with the poking interaction. This includes refining the precision and reliability of the interaction technique to ensure accurate app selection and minimize accidental selections. Implementing additional features, such as adding app names (instead of only the icons) and customization options such as grouping up apps based on user preference or topic could also be useful. A future user study could also compare the spherical layout with the traditional flat layout, to provide additional feedback on the necessary improvement that the sphere-based launcher may require.

REFERENCES

[1] 2022. *Microsoft: Direct manipulation with hands.* https://learn.microsoft.com/en-us/windows/mixed-reality/design/direct-manipulation

[2] 2022. *Microsoft: Point and commit with hands.* https://learn.microsoft.com/en-us/windows/mixed-reality/design/point-and-commit

[3] Mark Billinghurst, Raphael Grasset, and Julian Looser. 2005. Designing augmented reality interfaces. *ACM SIGGRAPH Computer Graphics* 39 (02 2005), 17–22. https://doi.org/10.1145/1057792.1057803

[4] Maite Frutos-Pascual, Chris Creed, and Ian Williams. 2019. Head Mounted Display Interaction Evaluation: Manipulating Virtual Objects in Augmented Reality. In *Human-Computer Interaction – INTERACT 2019: 17th IFIP TC 13 International Conference, Paphos, Cyprus, September 2–6, 2019, Proceedings, Part IV* (Paphos, Cyprus). Springer-Verlag, Berlin, Heidelberg, 287–308. https://doi.org/10.1007/978-3-030-29390-1_16