Proposal of Descriptive Approach for XR Device Sensor Data Handling and Compatibility

Yeseul Son
VR/AR Research Center
Korea Electronics Technology Institute
Seoul, Korea
yeseul0323@keti.re.kr

Yoonmo Yang
VR/AR Research Center
Korea Electronics Technology Institute
Seoul, Korea
yym064@keti.re.kr

Heeyoon Jeong
VR/AR Research Center
Korea Electronics Technology Institute
Seoul, Korea
nooyjx@keti.re.kr

Kwang-Soon Choi
VR/AR Research Center
Korea Electronics Technology Institute
Seoul, Korea
lenon@keti.re.kr

***Abstract*—This paper presents an approach to address the challenges of handling sensor data and compatibility issues arising from the rapid advancement of Extended Reality (XR) technology and the diverse range of XR devices in the market. In order to overcome the diversity and development challenges of XR devices, the concept of device profiling is introduced. This aims to provide XR application developers with a standardized method to define and manage sensor data for each device, thus offering convenience.**

An XR device profile consists of a device ID and sensor properties, where the latter includes sensor ID, type, data format, and characteristics. This information is expressed in JSON format, which offers various advantages, such as maintaining data structure consistency, hierarchical data representation, resolving language dependencies, readability, and ease of maintenance and customization. This approach of device profiling proposed in this paper offers a consistent and flexible way to handle the diverse sensor data of XR devices, thereby reducing developers' challenges and addressing compatibility issues. Furthermore, this profiling approach is expected to promote unity within the XR ecosystem and facilitate faster development and innovation in XR technology. We anticipate that future research and collaboration will further enhance sensor data management in XR devices and improve the developer experience.

Keywords—Extended Reality (XR), Sensor Data Processing, Compatibility Issues, Device Profiling, Developer Experience

I. INTRODUCTION

The digital technology innovation of the modern era is radically redefining our experiences and interactions. Among these innovations, Extended Reality (XR) technology is rapidly advancing, presenting new possibilities for transformative experiences and fostering new cultures and industries. XR encompasses various forms such as Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), merging real and virtual spaces. It is driving innovative applications across diverse fields like industry, education, entertainment, and exerting a wide-ranging impact on society[1-6]. However, the rapid advancement of XR technology and the proliferation of diverse XR devices have brought forth persistent challenges. The variation in sensor data formats, types, and characteristics across XR devices leads to significant hurdles for developers. Moreover, ensuring compatibility and consistency across different platforms remains a substantial concern.

To address the challenges arising from the diversity and complexity of XR devices, research efforts have emerged. A notable example is the Khronos Group's OpenXR initiative[7]. OpenXR, an open-source standard, aims to standardize development and compatibility within the XR domain, including VR and AR. It was established through collaboration among XR device manufacturers, software developers, and platform providers, aiming to address the fragmentation and compatibility issues in the XR ecosystem.

OpenXR provides standardized APIs and interfaces for interactions between XR devices and platforms. It enables developers to create applications consistently across various devices, allowing users to enjoy uniform experiences across different XR devices. However, although OpenXR provides some level of abstraction for XR device sensor data, it does not comprehensively address the specific characteristics of all sensors[8]. The abstraction of sensor data for certain devices may lack flexibility for developers due to its constrained nature. Furthermore, while OpenXR supports abstraction for certain sensor data, other data may need to be handled through device-specific extensions, making it challenging for developers to customize sensor data according to their needs.

The objective of this study is to address the diversity of XR devices and the challenges of development by utilizing the concept of device profiling. Device profiling involves defining and managing sensor data for each XR device through standardized methods, providing convenience to XR application developers. Through this approach, we aim to mitigate the difficulties arising from differences between XR devices, resolve platform dependency issues, and ensure compatibility and consistency across various platforms.

Overall, the study aims to leverage device profiling as a powerful tool to manage and utilize sensor data by reflecting the diversity and characteristics of XR devices, ultimately offering specialized developer experiences and precise control over sensor data. This is anticipated to positively impact the efficiency and performance enhancement of XR application development.

II. ANALYSIS OF SENSORS IN XR DEVICES

A. Analysis of XR Device Functionalities

To comprehend the sensors integrated into XR devices, an analysis of the functionalities offered by XR devices was conducted. The essential features that XR devices require to provide users with immersive experiences include:

- Display Functionality: Displaying virtual reality or augmented objects to users.
- Environment Perception and Position Tracking: Detecting the user's surroundings to provide realistic sight and experience.
- User Interaction and Input: Enabling head tracking to align the display with the user's movement and supporting input through controllers, hand gestures, and voice recognition.
- Environmental Audio: Enhancing auditory experiences by utilizing built-in microphones and speakers.
- Interaction between Virtual and Real Spaces: Detecting collisions when moving in virtual spaces and providing interactions like object movement between virtual and real environments.

B. Analysis of XR Device Sensors

In order to facilitate the functionalities outlined in the section II.A, the following sensors are necessary:

- Display Module: Primarily used in XR devices like head-mounted displays (HMDs), compact display modules improve visibility. They might be combined with ambient light sensors for display brightness adjustment.
- Cameras: Essential for environmental perception, position tracking, user interaction, and input. Cameras process captured images to recognize the surroundings, determine the user's current position, and track hand movements, gestures, and eye movement. Devices may use specialized cameras for specific purposes, often complemented by physical features like IMUs and depth sensors to enhance tracking accuracy.
- IMU Sensors: Comprising accelerometers, gyroscopes, and magnetometers, IMU sensors track the device's movements and orientation. They facilitate environmental perception, position tracking, and even head tracking when combined with cameras.
- Depth/Distance Sensors: Measuring distances from the device to surrounding objects, these sensors include stereo cameras, Time-of-Flight (ToF) Lidar, infrared (IR), lasers, and ultrasonic sensors. They are employed either independently for collision prevention or combined with cameras, as explained earlier, to enhance tracking precision.
- Controllers: These sensors detect hand gestures and input. They include buttons, touchpads, joysticks, and can even employ alternatives like voice recognition or hand gesture recognition based on the device.
- Microphones: Used to recognize user voice commands or capture ambient sounds.
- Speakers: Designed to provide auditory experiences to users. Depending on the device, they might even create immersive soundscapes through array configurations.

TABLE 1 summarizes the XR device sensors used for each of the functionalities described in Sections A and B. It categorizes XR device functionalities into essential and additional features and presents the primary sensors used for each function along with any additional sensors commonly used in conjunction.

TABLE I. SENSORS USED FOR XR DEVICE FUNCTIONALITIES

Functionality		**Main sensor**	**Additional sensor**
essential function	display	display module	light sensor
	environment perception and position tracking	camera	IMU depth sensor distance sensor
	user interaction and input	camera microphone controller	
optional function	environmental audio	speaker	microphone
	interaction between virtual and real spaces	distance sensor IMU	camera

III. XR DEVICE PROFILE

A. XR Device Profile Structure

Through the analysis of XR device sensors conducted in Chapter II, it was observed that the types and quantities of sensors included in XR devices vary significantly.

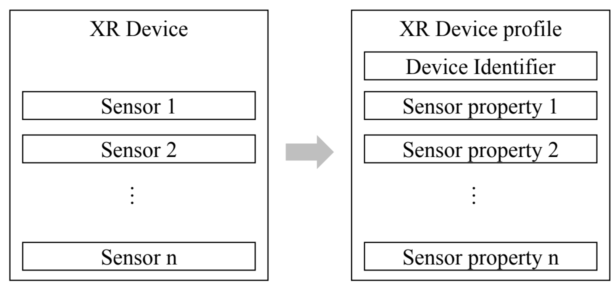

Figure 1. Structure of XR device and the device profile of it

To address this diversity without limiting the unique functionalities of devices, we have introduced the concepts of "Device Profiling" and "Sensor Property" to allow developers flexible utilization of sensor data from devices. For intuitive representation, we have designed the physical structure of XR devices and the grammatical structure of XR Device Profiles to be identical, as shown in Fig. 1.

A Device Profile consists of a Device ID for identification and Sensor Properties containing information about the sensors attached to the device. Since Sensor Properties can be added for each sensor attached to the device, there are no limitations to describing all sensors possessed by the device.

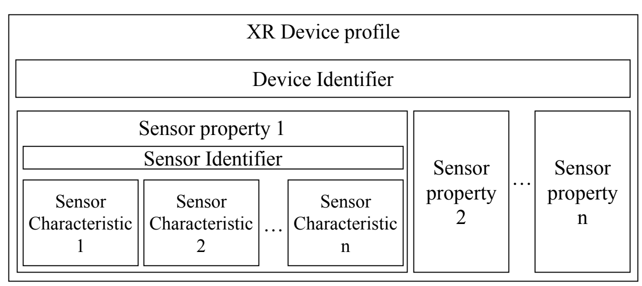

Figure 2 Structure of sensor properties within a device profile

Figure 2 depicts the structure of Sensor Properties within a Device Profile. As seen in Fig. 2, Sensor Properties include an ID for sensor identification, and they can encompass various characteristics such as sensor type, data format, and sensor parameters. The configuration of sensor characteristics varies depending on the sensor type.

B. *XR Device Profile Descriptive Approach*

We propose using JSON format, utilizing key-value pairs, to describe all sensors composing a device and to provide methods for obtaining characteristics and data of each sensor through Device Profiles and Sensor Properties. Utilizing JSON format for describing Device Profiles offers the following advantages:

1) Maintaining Data Structure Consistency: JSON objects consist of key-value pairs, where keys are strings, and values can be strings, integers, floats, objects, arrays, booleans, null, and various data types. Hence, it becomes possible to express diverse values associated with various sensors in a consistent manner.

2) Hierarchical Data Representation: JSON arrays can include other arrays or objects, allowing for the representation of the physical/logical hierarchy of sensors within XR devices.

3) Resolving Language Dependencies: Being independent of specific programming languages, JSON enables data exchange in various environments.

4) Readability: Intuitive and concise expressions enhance ease of understanding and use for developers.

5) Maintenance and Customization Ease: The flexible data structure facilitates modifications, contributing to better maintenance and customization aspects.

These advantages align well with our ultimate goal of resolving platform dependencies and providing convenience to developers. Fig. 3 and Fig. 4. illustrate an example of an XR device described using the proposed profiling approach in this paper. It demonstrates the ability to describe all sensors attached to an XR device regardless of sensor types or data formats, and showcases the intuitive and concise nature of expression achievable through this method.

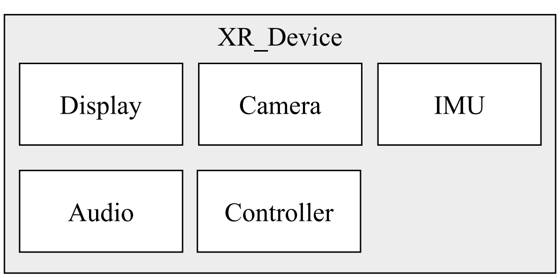

Figure 3. Example of XR device structure


```json
{
    "device_id": "XR_Device_123",
    "sensors": [
    {
        "sensor_id"     : 0,
        "sensor_type"   : "Display",
        "data_type"     : "Image",
        "properties"    : {
            "resolution"  : "1920x1080",
            "refresh_rate": "90Hz"
        }
      },
    {
        "sensor_id"     : 1,
        "sensor_type"   : "Camera",
        "data_type"     : "Image",
        "properties": {
            "resolution"    : "1280x720",
            "field_of_view" : "110°"
        }
    },
    {
        "sensor_id": 2,
        "sensor_type": "IMU",
        "data_type": "Orientation",
        "properties": {
            "gyroscope_range"     : "±2000°/s",
            "accelerometer_range" : "±16g"
        }
    },
    {
        "sensor_id": 3,
        "sensor_type": "Audio",
        "data_type" : "Sound",
        "properties": {
            "microphone_count" : "2",
            "speaker_count"    : "2"
        }
    },
    {
        "sensor_id": 4,
        "sensor_type": "Input Controller",
        "data_type": "Motion and Buttons",
        "properties": {
            "button_count": "8",
            "motion_tracking": "6DOF"
        }
    }
    ]
}
```


Figure 4. Example of XR Device Profile Description for the Illustrated Device in Fig. 3

IV. Conclusion

In this paper, we propose an approach to address the challenges of handling sensor data and compatibility issues arising from the rapid advancement of Extended Reality (XR) technology and the proliferation of various XR devices. To overcome the diversity of XR devices and the difficulties in development, we introduce the concept of device profiling. Through this, we aim to provide a standardized method for defining and managing sensor data for each device, offering convenience to XR application developers.

The XR device profile consists of a device ID and sensor properties, where sensor properties encompass identifiers, types, data formats, and characteristic values of sensors. This information is represented using JSON format, providing various advantages. JSON format offers benefits such as consistent data structure, hierarchical data representation, language independence, readability, and ease of maintenance and customization, all of which significantly contribute to developers' convenience.

The proposed device profiling approach in this paper presents a method to handle diverse sensor data of XR devices consistently and flexibly, alleviating developers' challenges and addressing compatibility issues. Moreover, this profiling approach is expected to promote unity within the XR ecosystem and facilitate the accelerated growth and innovation of XR technology. We anticipate further research and collaboration to enhance sensor data management in XR devices and improve the developer experience in the future.

ACKNOWLEDGMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00962, Development of industrial XR framework for smart manufacturing)

REFERENCES

[1] Ziker, Cindy, Barbara Truman, and Heather Dodds. "Cross reality (XR): Challenges and opportunities across the spectrum." Innovative learning environments in STEM higher education: Opportunities, challenges, and looking forward (2021): 55-77.

[2] Vasilchenko, Anna, et al. "Collaborative learning & co-creation in XR." Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems. 2020.

[3] Doolani, Sanika, et al. "A review of extended reality (xr) technologies for manufacturing training." Technologies 8.4 (2020): 77.

[4] Fast-Berglund, Åsa, Liang Gong, and Dan Li. "Testing and validating Extended Reality (xR) technologies in manufacturing." Procedia Manufacturing 25 (2018): 31-38.

[5] Voštinár, Patrik, Dana Horváthová, and Martin Bako. "XR Applications for the Education and Entertainment of Hospitalized Children." DIVAI 2022 (2022).

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982.

[7] The Khronos® Group Inc. (2023, 08 07). OpenXR. Available: https://www.khronos.org/openxr/.

[8] The Khronos® Group Inc. (2023, 08 07). OpenXR document. Available: https://registry.khronos.org/OpenXR/#apispecs