Received 4 December 2023, accepted 18 January 2024, date of publication 23 January 2024, date of current version 31 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357549

RESEARCH ARTICLE

Augmented Reality Creation Platform Based on Graphical Programming

JINFANG LI[1], YONGXI LIU[1], LIBAO XIAO[1], JIAHONG CAI[1], HANWU HE[1,2], AND XIAOWEI ZHANG[1]
[1]School of Electromechanical Engineering, Guangdong University of Technology, Guangzhou 510000, China
[2]Guangdong Polytechnic of Industry and Commerce, Guangzhou 510000, China

Corresponding author: Xiaowei Zhang (zxwei@gdut.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004902, and in part by the Guangzhou Science and Technology Project under Grant 2016A040403108.

ABSTRACT The development of conventional augmented reality (AR) works typically involves complex environment setup, programming development, high costs, and long development cycles, requiring developers to possess certain technical expertise and facing a high entry barrier. To address these challenges and reduce the difficulty of AR work development, this paper proposes a method for creating AR works based on graphical programming. A prototype system is developed to facilitate visual creation. The proposed method involves importing models and AR markers to construct the elements of the work. Subsequently, model parameter variables are designed to enable graphical programming and establish data associations between the work's elements and the logic function algorithm. Additionally, the front-end page layout of the work can be conducted, and the binding of front-end buttons to back-end graphical programming trigger events can be realized. The prototype system is developed using the AR-Foundation and Unity3D engine. Several case studies are presented to demonstrate the feasibility and versatility of this AR creation method.

INDEX TERMS Augmented reality, creation platform, graphical programming, visual creation.

I. INTRODUCTION

AR is a technology that integrates virtual objects into the real world, making it a crucial field in the new generation of artificial intelligence. AR offers real-time, interactive, and immersive experiences by presenting computer-generated virtual objects as if they exist in the physical environment. It finds wide applications in various fields such as industrial assembly [1], [2], healthcare [3], game [4], navigation [5], and education [6]. With the widespread adoption of mobile devices and 5G networks, the demand for resources and applications in mobile AR is increasing, emphasizing the importance of efficient and convenient AR work creation [7].

However, there are several challenges in the development and design of AR works and systems. Firstly, it requires a significant amount of mathematical, computer graphics, and programming knowledge, making specialized technical expertise necessary for development. This limits the participation of individuals without prior development experience in AR work creation [8], [9]. Secondly, different package

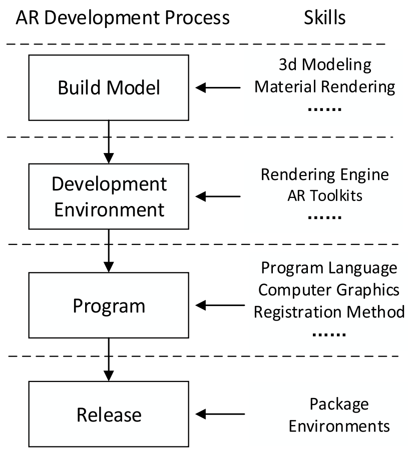


FIGURE 1. The conventional AR development process.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaogang Jin.

environments and AR toolkits need to be chosen based on the target platform [10]. For example, ARCore is commonly used for Android platforms, while ARKit is used for iOS platforms [11]. Additionally, many functional modules of AR works do not need to be reinvented during the development process, such as registration methods [12]. The conventional AR development process is illustrated in Fig. 1

Therefore, the originally straightforward process of designing AR works becomes challenging due to the skill requirements and complexity of the development environment, and the process can be simplified. Consequently, there is a demand for the development of a unified and convenient AR platform in the current AR technology landscape. The ISMAR's Ten-Year Impact Award Report on AR research [13] highlights the development of AR creation tools (AR platform) as one of the hot topics in AR research.

Among the existing AR platforms, Li et al. [14] developed AR works based on route maps and complex logical expression methods, defining AR works as "marker-model-event-behavior." However, they overlook the fact that triggering behaviors can be defined based on the associated states between multiple AR models. Researchers Yang et al. [15] and Cavallo and Forbes [16] have also conducted relevant studies. Their common features include non-textual programming and visual design, focusing on the display of AR models and simple "move, rotate, and scale" functionalities. Apple's AR platform, Reality Composer [17], provides model behavior interaction design through setting trigger conditions, adding action sequences, and defining the flow of "affected objects," allowing for more flexible AR works design. However, the trigger conditions for model behaviors are limited, and the triggering events are only based on time sequences. Meta's Meta Spark [18] and Snapchat's Lens Studio [19] enable rapid development of AR filters for video calls and short videos, with programming development for interaction design. Unity MARS [20], provided by Unity, is a development tool that facilitates AR works programming development for Unity users.

From these various AR platforms, it is evident that most platforms are developed based on the concept of AR animation design, focusing on showcasing AR models rather than constructing AR works with strong logical structures.

Therefore, the AR Creation Platform combines graphical programming from the field of visual implementation. Leveraging the convenience, rapidity, and intuitiveness in design, it applies graphical programming techniques to the field of rapid AR development.

The platform proposes a new method for AR works design based on graphical programming. It defines two fundamental data structures: AR elements, encompassing models and their corresponding AR registration methods, and Parameters, including adjustable variables related to the model (e.g., position, rotation, scaling ratio, animation, and child object information), integrated into graphical programming blocks.

Expanding on these data structures, the method for AR works design is elucidated: users choose the appropriate AR elements and visually design logical statement blocks that involve AR elements and Parameters. Subsequently, a seamless connection is established between the front-end interface and the back-end graphical programming logic, real-time response algorithms and trigger response algorithms within AR works can be realized. Once AR works constructed, the works can be previewed and debugged. Finally, the entire works can be saved and exported. The entire design process is presented through a visual interface, eliminating the need for complex development environments and programming skills while achieving AR works.

II. COMPOSITION OF THE AR CREATION PLATFORM

A. DEVELOPMENT TOOLS OF THE AR CREATION PLATFORM

When it comes to AR software development, commonly employed development tools include the Unity3D and Unreal Engine 4 gaming engines [21]. Concurrently, prevalent AR Software Development Kits (SDK) encompass AR-Foundation, Vuforia, ARCore, ARKit, among others [22].

The AR Creation Platform is developed using Unity3D and programmed in C#. The AR SDK used in the platform is Unity3D's AR-Foundation.

The Unity3D is an AR experience development engine that supports various platforms, including PC, mobile devices, and AR devices [23]. In the field of AR, Unity3D provides the AR-Foundation SDK as a development tool. It is widely employed in the development process of AR applications.

The AR-Foundation is a collection of SDKs such as ARKit and ARCore, providing developers with a unified interface that automatically selects the appropriate underlying SDK version based on the target platform for publishing [24].

B. MODULES OF THE AR CREATION PLATFORM

Based on the works design process, the platform incorporates the following modules: AR Asset Manager, AR Editor, and AR Debugger, as shown in Fig. 2.

The AR Asset Manager serves as the interface for importing resources (including AR markers and models). It utilizes AR-Foundation to register models and construct the AR elements of this platform.

The AR Editor is the core module of the platform. It allows model parameter editing as variables and animation design for the created AR elements. It employs graphical programming to design the interaction logic of AR and binds the front-end presentation with the back-end data, establishing the connection between the two.

The AR Debugger provides users with an AR environment with screen interaction capabilities. It allows for the verification of model registration effects, interaction logic and serializes the AR work after debugging.

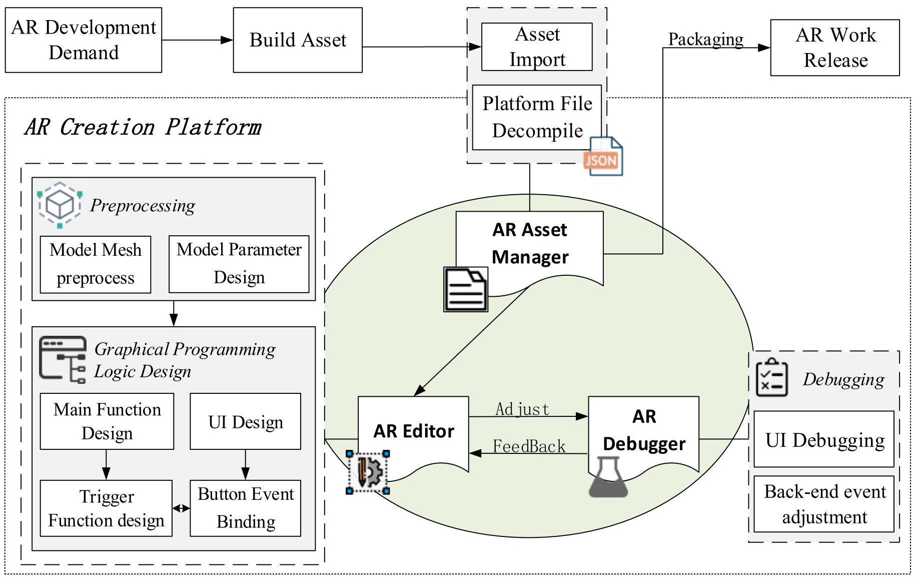


FIGURE 2. AR creation platform development process.

III. AR ELEMENTS

A. COMPOSITION OF AR ELEMENTS

The AR Creation Platform begins with the creation and management of AR elements, which include models and registration methods. Within this platform, the supported file formats for AR models are FBX and OBJ. FBX format can be read and written by 3D modeling software such as 3Dmax and Maya, supporting various materials and animations [25]. It is widely used as a data exchange format. On the other hand, OBJ format allows for import and export in all modeling software. The registration methods encompass plane detection and image-based recognition. Plane detection does not require pre-imported markers, while image-based recognition necessitates binding to markers. The supported file formats for markers are common JPG and PNG formats, with a minimum pixel size of 512 × 512. The images should possess distinct feature points to ensure the accuracy of image-based recognition.

B. CONSTRUCTION OF AR ELEMENTS

The construction of AR elements primarily involves setting up registration methods for models and model initialization.

1) MODEL REGISTRATION METHOD

The AR SDK(AR-Foundation) offers functionalities for image-based recognition and plane detection. Image-based recognition relies on capturing the position information of recognized AR markers' features [26], while plane detection emphasizes the identification of plane in the actual environment [27].

Specifically, when creating AR work that involve the recognition of multiple AR marker and require specific logical controls based on these AR markers, image-based recognition is the preferred choice. However, in scenarios where an AR work is desired without the need for complex marker-based logic, plane detection is more suitable because it is still a challenge to extract robust features in weakly texture scenes [28].

Hence, considering the characteristics of image-based recognition and aiming to enhance the recognition accuracy and stability of imported AR markers, it is crucial, when utilizing the image-based recognition tracking feature, to ensure that the imported AR markers possess an adequate number of feature points and to avoid importing markers closely resembling existing ones. Therefore, this platform employs an algorithm based on ORB feature point detection to perform feature matching between the currently imported AR marker and all markers in the existing marker library. If the number of matched points between the imported AR marker and an existing marker exceeds a threshold, it indicates similarity between the imported AR marker and the existing ones and stop the import image process. The ORB feature point algorithm is implemented based on the OpenCV library, and the specific process is illustrated in Fig. 3.

2) MODEL INITIALIZATION

Due to potential deviations in coordinate axes and model positions that may occur during the modeling process in many FBX format files, discrepancies can arise between the coordinate system used for mesh rendering (Rendering Space) and the actual model coordinate system (Pivot Space) when dynamically importing them into engines like Unity3D. Hence, this platform incorporates an initialization process to ensure that the center of the mesh rendering bounding box aligns with the actual coordinate point of the model upon import.

First, a minimum AABB (Axis-Aligned Bounding Box) is added to the model. Then, the coordinates of the rendering

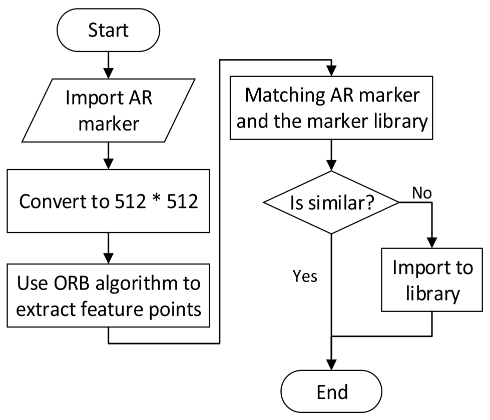

Start
Import AR marker
Convert to 512 * 512
Use ORB algorithm to extract feature points
Matching AR marker and the marker library
Is similar?
No
Yes
Import to library
End

FIGURE 3. Import image process.

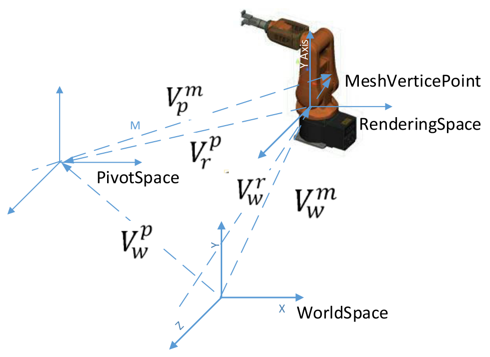


FIGURE 4. Coordinate transformation relation.

center point and the actual model coordinates are aligned. In Unity3D, the coordinates of the model's mesh points are based on its own actual coordinate system. To address this, the coordinates are transformed into world coordinates, followed by an offset of the mesh vertex coordinates, and finally, they are transformed back into the model's local coordinate system.

$$P_w^m = M_w^p \cdot P_p^m \tag{1}$$

where $\mathrm{P}_\mathrm{w}^\mathrm{m}$ and P_p^m represent the positions and scale ratios of mesh points in the world coordinate system and the local coordinate system, respectively.

$$\begin{cases} P_p^m = \left[x_p^m, P_p^m, P_p^m, 1\right]^T = \left[V_p^m, 1\right]^T \\ P_w^m = \left[x_w^m, y_w^m, z_w^m, 1\right]^T = \left[V_w^m, 1\right]^T \end{cases} \tag{2}$$

Once $\mathrm{P}_\mathrm{w}^\mathrm{m}$ is obtained, according to Fig. 4, each mesh point's coordinates are offset by V_r^p, where V_r^p is the vector between the model's local origin coordinates and the center coordinates of the minimum bounding box:

$$\begin{cases} V_m^{w^*} = V_m^w + V_r^p \\ P_w^{m^*} = \left[V_w^{m^*}, 1\right]^T = \left[x_w^{m^*}, y_w^{m^*}, z_w^{m^*}, 1\right]^T \end{cases} \tag{3}$$

Since the re-rendering process of mesh points is calculated based on the position data of mesh points in their local coordinates, therefore:

$$P_p^{m^*} = M_p^{*w} \cdot P_w^{m^*} \tag{4}$$

where M_p^{*w} is the transpose of matrix M_w^p.

After completing the initialization process of the AR model, all position coordinates, rotation values, and scaling ratios of the model are unified. This facilitates the subsequent AR design methods in handling the model and avoids any visual design impact caused by variations in the coordinate axes of imported models.

IV. AR WORKS CREATION METHOD BASED ON GRAPHICAL PROGRAMMING

In order to provide users with an intuitive way to create interactive interfaces and logic within AR works, this platform proposes a method for AR works creation based on graphical programming. This method consists of two parts: front-end design and back-end design based on graphical programming. Button-triggered events are used to establish the connection between the front-end and back-end, and trigger events are constructed through graphical programming to manage the control between models in AR works.

A. FRONT-END DESIGN

The front-end design is based on the visual representation of the front-end interface, where the designed UI modules are categorized into three types: images, text, and buttons (Fig. 5). Users can arrange the interface and modify attribute parameters through drag-and-drop operations to achieve front-end design. The parameter variables for each UI module during design include UI name, UI size, UI center, and UI layer.

The properties setting panel on the right-hand side is used for configuring buttons, and the dropdown menu at the bottom right allows users to select pre-defined trigger methods from the graphical programming, it is the "CHECK" event, which is bound to the method named "CHECK" in graphical programming). This enables the binding of buttons with events, establishing the connection between the front-end and back-end.

B. BACK-END DESIGN BASED ON GRAPHICAL PROGRAMMING

To provide more flexible control methods in AR works for developers without programming experience, the AR Creation Platform implements back-end design using graphical programming. Graphical programming involves converting basic data modules such as variables, statements, and function blocks from conventional text-based programming into different types of "building blocks." These building blocks are then assembled and manipulated through simple operations

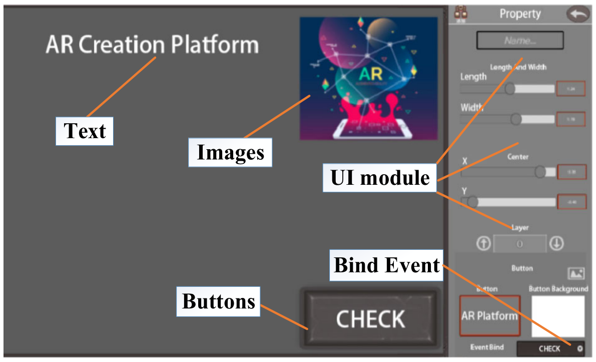


FIGURE 5. **Front-end interface design.**

such as block connection and filling to achieve programming effects and control the AR works.

1) AR ELEMENTS DATA MODULE

The control behavior of AR elements primarily involves the variation of model parameters. Therefore, data modules are constructed based on the parameters of the model (TABLE 1).

By designing graphical programming logic, custom changes can be made to the model's parameters, enabling control over the behavior and interaction of AR elements.

TABLE 1. **Model parameters.**

Parameters	Visible	Position	Rotation	Scale	Animation
Variable Type	Bool	Vector3	Vector3	Float	Bool

2) GRAPHICAL PROGRAMMING MODULE

Conventional procedural text-based programming languages typically include modules such as the Main Function, Trigger Method, Conditional Statement, Loop Statement, Assignment Statement, and Variables module. The platform combines these modules with AR elements modules to construct a lightweight graphical programming language for controlling the behavior of AR works. (TABLE 2)

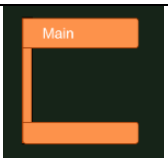
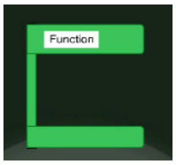
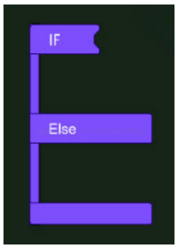
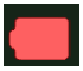
Users need to plan and design the behavior control logic for each element in the AR works before engaging in graphical programming. Using the graphical blocks, they can assemble the statements and perform variable assignments in a sequential manner. After testing and debugging the code using the AR debugger to ensure the desired effects are achieved, the back-end design is considered complete.

3) GRAPHICAL PROGRAMMING COMBINE RULES

During the design process, developers are aided by graphical elements such as size, shape, color, and drag-and-drop feedback to facilitate rapid programming design. The interaction method involves simple drag-and-drop connections, allowing developers to focus on the design process without worrying about syntax errors or other issues commonly associated with traditional text-based programming. Different shapes and colors are used to differentiate module connections and containment relationships within the programming blocks.

For example, the Main Function and Trigger Function serve as containers for all other statement modules, so Conditional Statement, Loop Statement, and Assignment Statement have square shapes. The variable types include Expression Module and Variable. Operators used for expressions' left and right values are outlined in TABLE 3.

TABLE 2. **Graphic programming statement module.**

Statement Module	Icon	Meaning
Main Function		Function executed at the beginning
Trigger Method		Bind front-end button events
Conditional Statement		Execute different logic based on conditions
Loop Statement		Loop execute logic based on conditions
Assignment Statement		Assignment variables
Expression Module		Calculation between variables
Judgement Module		Equivalence operations between variables
Variable		Model parameter variable and basic variables

In terms of calculations, the distance between three-dimensional vectors is computed using the Vector3 Arithmetic, while rotation values are determined by calculating the angle between two vectors (ranging from 0 to 180 degrees). Additionally, when constructing expressions, both the left and right values can be iterated with new Expression Module and Variable (Fig. 6).

TABLE 3. **Operator category.**

Operator	+,-,×,÷	and	or	~	○
Bool	×	AND	OR	×	×
Float	Arithmetic Rules	×	×	×	×
Vector3	Vector3 Arithmetic	×	×	Calculate Distance	Calculate Rotation

*In the chart, '×' denotes the absence of this type of operation or calculation

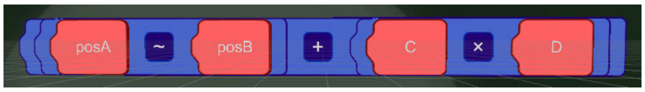


FIGURE 6. **Expression.**

The equivalent expression of the given expression is:

$$\sqrt{(a.x-b.x)^2+(a.y-b.y)^2+(a.z-b.z)^2}+cd \qquad (5)$$

where ‘PosA’ and ‘PosB’ are the coordinate values of the 3D vectors, ‘C’ and ‘D’ is a float type variable.

4) GRAPHICAL PROGRAMMING DATA PROTOCOL

In the AR Creation Platform, the data protocol plays a crucial role in the import and export functionality of files. The platform provides a set of protocols to validate file formats and perform error checking on file content, enabling cross-platform design. All data in graphical programming can be classified into two types: Statements and Variables. Statements include Assignment Statement and Logical Statement, while variables encompass model parameter variables and basic variables (Fig. 7).

Currently, the data communication protocols include XML, JSON, and binary serialization methods. Serialization of models and AR markers is best suited using byte arrays. Prior knowledge of the data packet’s size is necessary due to the limited loading and reading capacity of models with a high number of mesh vertices on mobile devices. Therefore, the binary serialization method is chosen. However, because tree-structured classes cannot be directly serialized by using binary serialization, the platform converts the statement tree structure in graphical programming into an array of serializable classes. When parsing the file, this array is then converted back into the original statement tree structure using a specific algorithm.

Input: Statement tree T.

Output: Ordered list of statements A.

Step 1: Create a statement queue Q and initialize an index variable Index. Enqueue the root node of T into Q and set Index to the number of child statements of the T root node.

Step 2: Check if the queue Q is empty. If not empty, perform the following steps. Otherwise, proceed to Step 3.

Step 2.1: Dequeue the top element from the queue Q. If the element is a logical statement, iterate through all its child statements, record their indices, and enqueue them into Q. Update Index to keep track of the indices of the first and last child statements of the top element.

Step 2.2: Add the top element to the ordered list of statements A.

Step 2.3: Return to Step 2 and continue.

Step 3: End the algorithm.

Considering the example of a graphical programming for a convex lens imaging AR works, we have the following graphical programming representation and the corresponding statement tree and statement linked list: (Fig. 8).

V. PLATFORM CASES

A. PLATFORM INTERFACE

The platform interface consists of three main modules: Model Editing Area, Menu Bar, and Model Information Area. The Graphical Programming Interface includes three modules: Graphics Module Area, Graphical Programming Area, and Model Matching Area. (Fig. 9).

The AR Creation Platform allows for cross-platform creation of AR works on PC, Android, and IOS devices. However, browsing AR works is only supported on Android and iOS devices. After creating a work on any device, the AR platform files can be exported and imported into another device for editing. Cross-platform effects are depicted in Fig. 10.

B. DESIGN OF CONVEX LENS IMAGING AR WORK

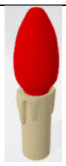
The work involves three AR elements: a candle, a convex lens, and a plane with an image. There are several requirements for the work:

a) When the model distance and AR marker distance change, the size of the image on the plane will adjust.

b) The image of the candle on the plane will only appear if all three elements (candle, convex lens, and plane) are present in the scene.

To meet these requirements, three AR elements are designed, and image-based recognition methods are used for registration. The corresponding relationships are shown in TABLE 4.

TABLE 4. **Construction of AR elements.**

AR Elements	Candle	Convex Lens	Plane (with image)
Model			
AR Marker			

Next, based on the algorithms, a graphical programming is constructed as shown in Fig. 8(a). First, the model parameter variables that need to be controlled are set, as shown in TABLE 5. These variables represent the parameters of the model that can be read or modified in the method design within the graphical programming.

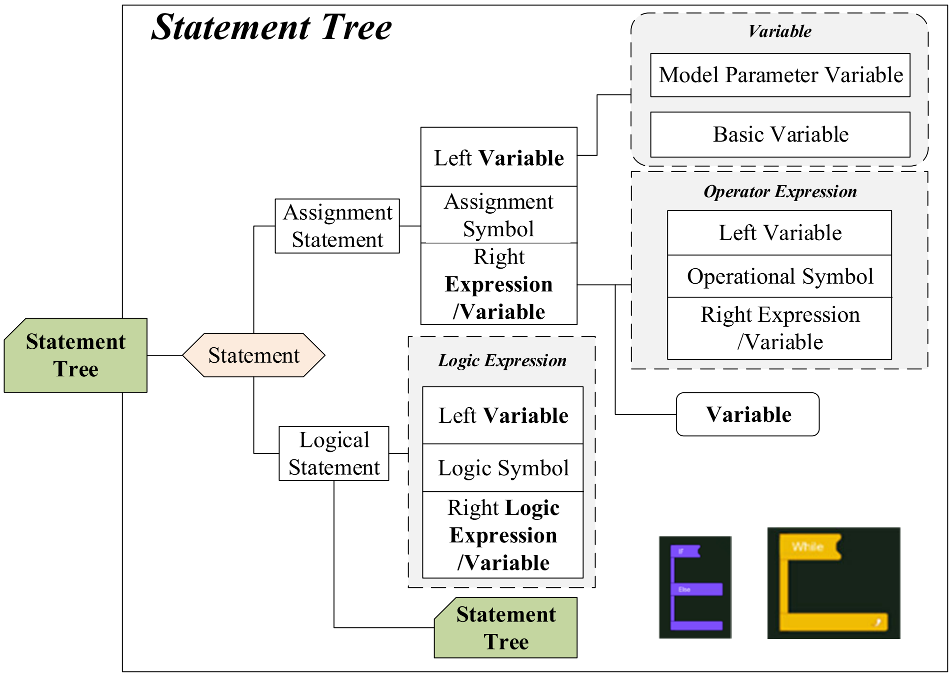


FIGURE 7. Statement tree programming progress.

TABLE 5. Convex lens imaging AR work variables.

Model	Candle	Convex Lens	Board	Image
Model Parameter	Visible Position	Visible Position	Visible Position	Visible Scale

When the distance between the candle and the mirror is large, the image becomes smaller. When the distance decreases, the image enlarges. If the mirror maker is missing in the scene, the image disappears. The specific AR effects of the hardware devices are displayed in Fig. 11.

C. DESIGN OF ROBOT INSTALLATION AR WORK

In the Robot Installation AR work, the goal is to showcase the layout of robots on the training platform using AR. Fig. 12 presents the specific AR effects of the hardware devices. This work allows users to run animation sequences of installing or dismantling. AR registration is achieved through plane detection. The algorithmic approach in graphical programming involves defining the starting point of the robot component's positional movement to define the animation of installation or dismantling. By clicking the "install" or "uninstall" buttons on the left and right sides of the screen, users can respectively browse the installation or dismantling process.

D. DESIGN OF SECTIONAL VIEW MODEL AR WORK

In the Mechanical Drawing course, the valve body model is a complex 3D model often used for teaching purposes in sectional view methods. This platform can be utilized to create and design a sectional view of the valve body. The specific AR effects of the hardware devices are displayed in Fig. 13.

By using image-based recognition as the registration method, the physical marker can be rotated to observe the valve body model from different angles and achieve the visual effect of a sectional view. The algorithm for displaying the sectional view of the valve body involves controlling the visibility of the valve body model's sub-objects. The algorithm is triggered by the design of the front-end buttons.

VI. AR PLATFORM COMPARISON

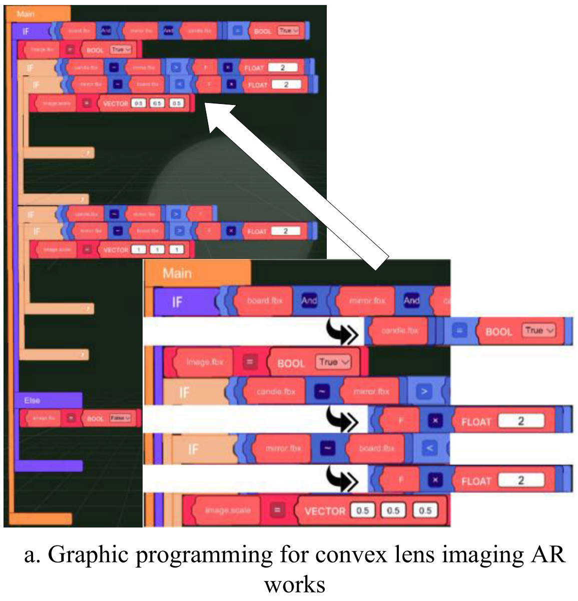
When compared to other AR platforms, as illustrated in TABLE 6, the AR Creation Platform offers several distinct advantages over simplified creation tools and AR platforms:

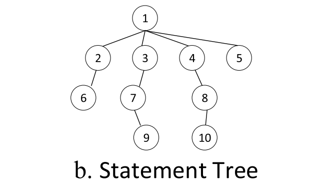
A. GRAPHICAL DESIGN APPROACH

The AR Creation Platform enables intuitive and graphical design, eliminating the need for users to master complex AR algorithms and programming techniques. This approach allows for a shorter development cycle and easier creation of AR experiences.

B. COMPREHENSIVE DESIGN CAPABILITIES

Unlike some other AR platforms that focus on specific aspects such as information presentation or AR filters, the AR Creation Platform provides comprehensive design





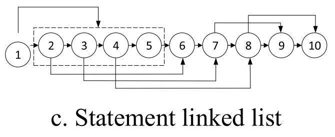


FIGURE 8. **Graphical programming data structures.**

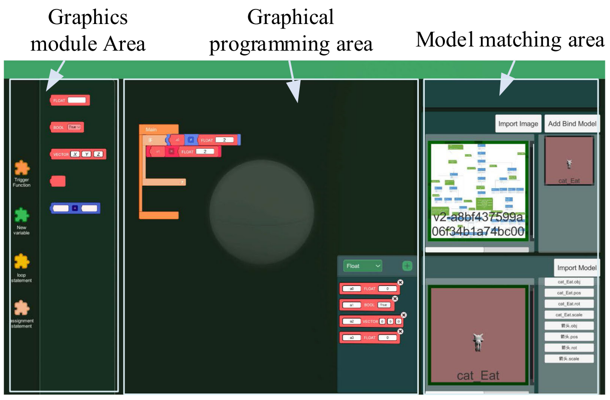


FIGURE 9. **Platform interface.**

capabilities. It allows users to design connections between models, create custom trigger events based on model parameters, and bind front-end buttons to back-end graphical programming functionality.

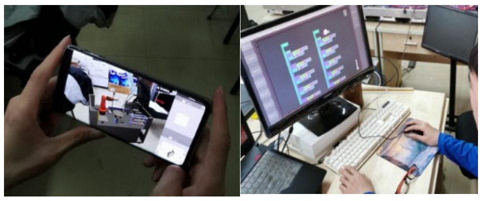


FIGURE 10. **AR design and browsing effects.**

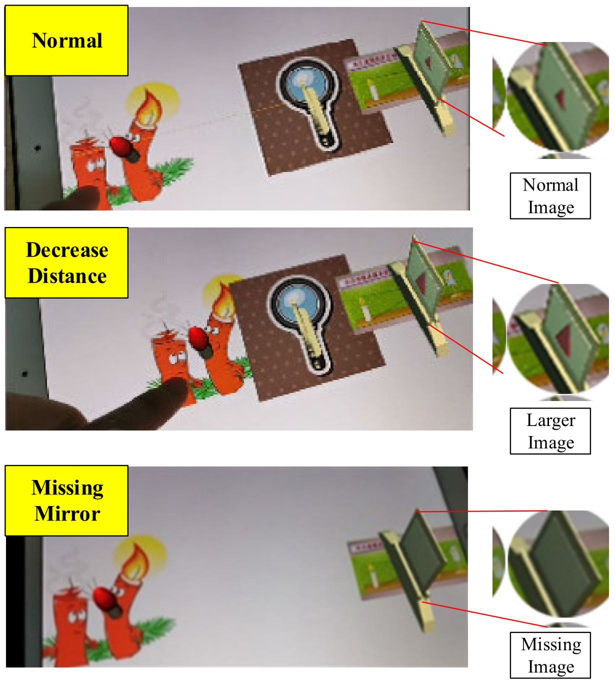


FIGURE 11. **AR convex lens experimental effect.**

C. CROSS-PLATFORM SUPPORT

The AR Creation Platform supports designing and publishing for multiple terminals, allowing users to create AR experiences for PC, Android, and iOS devices. This cross-platform compatibility provides flexibility and broader reach for AR works.

D. EASE OF USE FOR NON-PROFESSIONALS

The AR Creation Platform's zero-code approach and user-friendly interface make it accessible to non-professionals. Users do not require extensive skills to create AR works, distinguishing it from platforms like Unity MARS that target more advanced developers.

E. PRACTICAL DESIGN APPROACH

The design process of the AR Creation Platform, as demonstrated in multiple cases, proves to be practical and effective. It empowers users to design complex AR interactions and behaviors without the need for complex coding or resource-intensive asset generation.

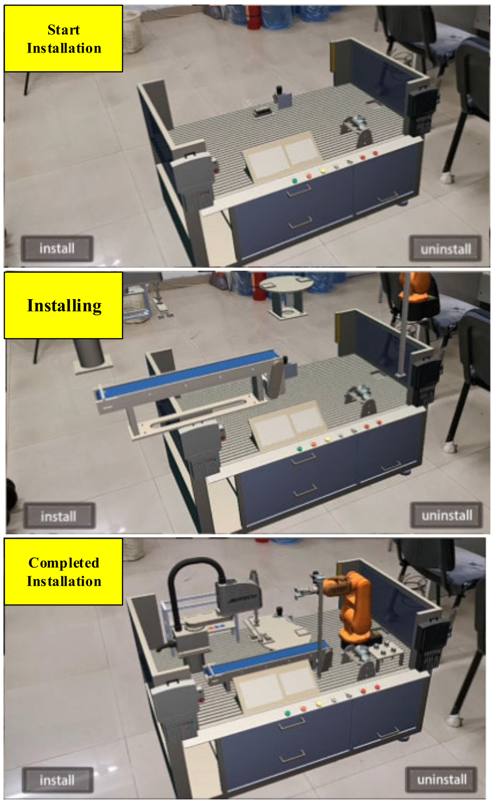


FIGURE 12. Industrial robot training platform layout AR works.

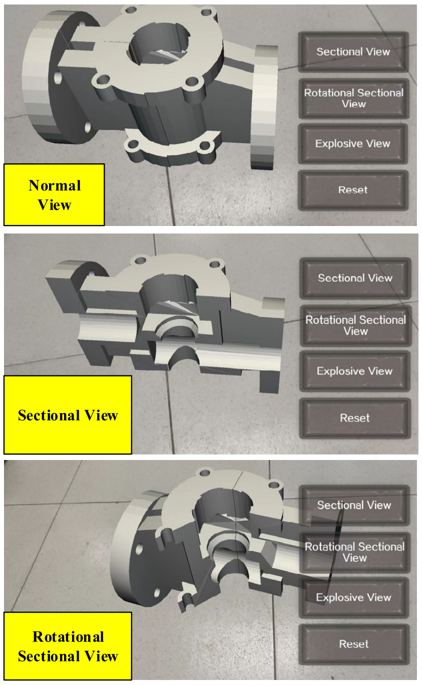


FIGURE 13. AR valve body model display.

Overall, the AR Creation Platform combines ease of use, comprehensive design capabilities, cross-platform support, and practical design approaches, making it a powerful tool for creating AR experiences with a wide range of applications.

TABLE 6. Multiple AR platform compare.

Comparator	AR Creation Platform	Reality Composer	Spark AR	Unity MARS
Trigger Events	Multiple	4	Programing	Programming
Sub-object Editing	Yes	No	Yes	Programming
Cross-platform Support	Yes	No	Yes	Environmental Setup
Front-end Design	Yes	No	No	Programming
Graphical Programming	Yes	No	No	No

VII. PLATFORM ASSESSMENT

To assess the AR Creation Platform, a group of participants was gathered to utilize the platform for completing AR works and provide feedback through a questionnaire (TABLE 7). The questionnaire aimed to evaluate the functionality, interactivity, and user experience of the platform's AR work creation method.

TABLE 7. Questionnaire content.

Title	Questions
1	Time taken to complete the convex lens imaging AR work
2	Time taken to complete the robot installation AR work
3	Time taken to complete the sectional view model AR work
4	Whether graphical programming was used during the development process
5	Whether there were any error prompts during the graphical programming design process
6	Whether platform improved efficiency in developing AR work
7	Whether platform was easy to operate without prior knowledge
8	Friendliness of the platform's development process interaction
9	Efficiency of the graphical programming design process
10	Difficulty level of the platform's AR work development process

The assessment involved 10 participants, all of whom were students with no prior text-based programming experience. The participants' ages ranged from 18 to 24, and they had some familiarity with AR technology before the evaluation.

Figure 14 illustrates the average time taken by participants to complete Work 1, Work 2, and Work 3. These AR works correspond to the well-functioning AR work with logic described in Sections B, C, and D of Chapter V. It can be observed that as participants gained more experience with the platform, they gradually became proficient in using it. Most participants were able to complete their works within 10 to 15 minutes, indicating that the platform's development process is efficient and allows for the creation of AR works with inherent logic.

Fig. 15 illustrates the statistical results of Questions 4, 5, 6,7, and 8, which assessed participants' usage, feedback, and the perceived difficulty of graphical programming. It is evident that the majority of participants provided positive evaluations of the platform, with more positive feedback than negative feedback. The occurrence of error

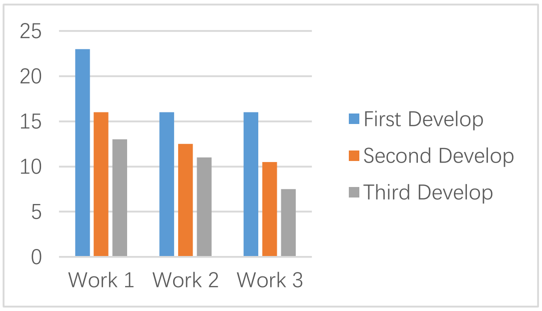


FIGURE 14. Average development time of three works(minutes).

prompts mentioned in Question 5 varied among participants due to different design situations. This demonstrates the robustness and user-friendliness of the platform to a certain extent.

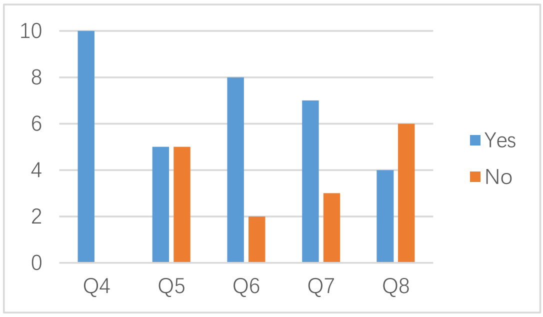


FIGURE 15. Platform usage evaluation.

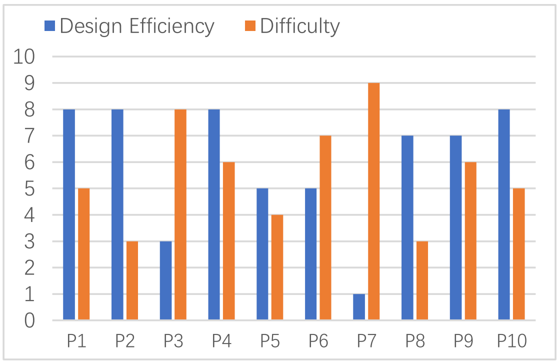


FIGURE 16. Design efficiency and ease.

Fig. 16 illustrates the statistical results of Questions 9 and 10, which aimed to investigate participants' evaluation of the design efficiency and the difficulty level of the design process. Both questions were scored on a range of 0 to 10, with higher scores indicating better evaluation. In terms of design efficiency, most participants gave higher scores, indicating a positive evaluation. Regarding the difficulty level, higher scores indicated a harder understanding of the platform's usage and design methods. The majority of participants provided higher scores for design efficiency compared to the difficulty level. This suggests that most participants considered the platform's design methods and implementation process to be meaningful.

VIII. CONCLUSION

A Method has been proposed for creating AR works based on graphical programming, resulting in the development of an AR creation platform.

The proposed method enables users to visually define the inherent logic and behavioral data between virtual objects in AR works, eliminating the need for AR expertise or programming skills. It facilitates the creation of AR works that go beyond simple animation design and offer enhanced interactivity. The developed cross-platform AR creation platform, implemented using Unity3D, significantly reduces the entry barrier, lowers development costs, and allows users to intuitively create AR works with simple interactivity and broad applicability. It holds substantial practical value.

Although the platform possesses general AR creation functionality, it lacks specific design modules and related resources for fields such as AR books and AR assembly/disassembly. Additionally, the platform lacks modeling capabilities, requiring users to create their own model resources. Future research could focus on the rapid construction of AR models to address these limitations.

Concurrently, AR work collects data about the user and the user's physical environment, which also includes other individuals. These intimate data can be collected, stored, and shared without others noticing [29]. Concerning privacy issues and the potential for misuse in AR work, the limitations of the AR creation platform are not sufficiently robust. Future efforts should enhance the platform's capability to restrict and guide privacy concerns during the creation process of AR work.

Overall, the AR creation platform and the graphical programming method introduced pave the way for accessible and user-friendly AR content creation, opening up new possibilities for creative expression and interactive experiences in the realm of AR.

REFERENCES

[1] W. Li, J. Wang, S. Lan, S. Li, S. Jiao, and M. Wang, "Content authoring of augmented reality assembly process," *Comput. Integr. Manuf. Syst.*, vol. 25, no. 7, pp. 1676–1684, Jul. 2019.

[2] X. Wang, S. K. Ong, and A. Y. C. Nee, "A comprehensive survey of augmented reality assembly research," *Adv. Manuf.*, vol. 4, no. 1, pp. 1–22, Mar. 2016, doi: 10.1007/s40436-015-0131-4.

[3] H. Yan and J. Guan, "The scalp localization system of neurosurgery based on augmented reality theory," *J. Biomed. Eng.*, vol. 36, no. 3, pp. 428–434, 2019, doi: 10.7507/1001-5515.201711061.

[4] J. M. Patricio, M. C. Costa, J. A. Carranca, and B. Farropo, "SolarSystemGO—An augmented reality based game with astronomical concepts," in *Proc. 13th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2018, pp. 1–3, doi: 10.23919/CISTI.2018.8399284.

[5] S. Rajeev, A. Samani, K. Panetta, and S. Agaian, "3D navigational insight using AR technology," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Nov. 2019, pp. 1–4, doi: 10.1109/HST47167.2019.9033006.

[6] X. Li, J. Huang, F. Tian, J. Liu, H. Wang, and G. Dai, "A survey on storytelling methods in modern interaction technology," *J. Comput.-Aided Des. Comput. Graph.*, vol. 10, pp. 349–363, Jan. 2019.

[7] X. Gao, H. An, and W. Chen, "A survey on mobile augmented reality visualization," *J. Comput.-Aided Des. Comput. Graph.*, vol. 30, no. 1, pp. 1–8, Jan. 2018.
[8] M. Nebeling and M. Speicher, "The trouble with augmented reality/virtual reality authoring tools," in *Proc. IEEE Int. Symp. Mixed Augmented Reality Adjunct (ISMAR-Adjunct)*, Munich, Germany, Oct. 2018, pp. 333–337, doi: 10.1109/ISMAR-Adjunct.2018.00098.
[9] M. Puggioni, E. Frontoni, M. Paolanti, and R. Pierdicca, "ScoolAR: An educational platform to improve students' learning through virtual reality," *IEEE Access*, vol. 9, pp. 21059–21070, 2021, doi: 10.1109/ACCESS.2021.3051275.
[10] J. Linowes and K. Babilinski, *Augmented Reality for Developers: Build Practical Augmented Reality Applications With Unity, ARCore, ARKit, and Vuforia*. Birmingham, U.K.: Packt Publishing, 2017.
[11] L. M. Ortega, J. M. Jurado, J. L. L. Ruiz, and F. R. Feito, "Topological data models for virtual management of hidden facilities through digital reality," *IEEE Access*, vol. 8, pp. 62584–62600, 2020, doi: 10.1109/ACCESS.2020.2984035.
[12] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017, doi: 10.1109/ACCESS.2017.2698164.
[13] K. Kim, M. Billinghurst, G. Bruder, H. B. Duh, and G. F. Welch, "Revisiting trends in augmented reality research: A review of the 2nd decade of ISMAR (2008–2017)," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 11, pp. 2947–2962, Nov. 2018, doi: 10.1109/TVCG.2018.2868591.
[14] Q. Li, H. He, and Y. Wu, "Research and implementation of authoring platform for augmented reality," *Comput. Eng. Des.*, vol. 39, no. 10, pp. 3296–3300, 2018.
[15] Y. Yang, J. Shim, S. Chae, and T.-D. Han, "Mobile augmented reality authoring tool," in *Proc. IEEE 10th Int. Conf. Semantic Comput. (ICSC)*, Feb. 2016, pp. 358–361, doi: 10.1109/ICSC.2016.42.
[16] M. Cavallo and A. G. Forbes, "CAVE-AR: A VR authoring system to interactively design, simulate, and debug multi-user AR experiences," in *Proc. IEEE Conf. Virtual Reality 3D User Interface (VR)*, Mar. 2019, pp. 872–873, doi: 10.1109/VR.2019.8798148.
[17] Apple. *Reality Composer*. Accessed: Aug. 10, 2023. [Online]. Available: https://apps.apple.com/us/app/reality-composer/id1462358802
[18] Meta. *Meta Spark*. Accessed: Aug. 11, 2023. [Online]. Available: https://spark.meta.com/
[19] Snapchat. *Lens Studio*. Accessed: Aug. 12, 2023. [Online]. Available: https://ar.snap.com/lens-studio
[20] W. Tabone, Y. M. Lee, N. Merat, R. Happee, and J. De Winter, "Towards future pedestrian-vehicle interactions: Introducing theoretically-supported AR prototypes," in *Automotive User Interfaces and Interactive Vehicular Applications*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 209–218, doi: 10.1145/3409118.3475149.
[21] W. Dokonal, M. Knight, and E. A. Dengg, "New interfaces–old models," in *Proc. eCAADe*, Sep. 2015, pp. 101–106, doi: 10.52842/conf.ecaade.2015.1.101.
[22] E. M. Bourhim and A. Akhiate, "Augmented reality SDK's: A comparative study," in *Intelligent Systems Design and Applications* (Lecture Notes in Networks and System). Cham, Switzerland: Springer, 2021, pp. 559–566, doi: 10.1007/978-3-030-96308-8_52.
[23] X. Hu, F. R. Y. Baena, and F. Cutolo, "Alignment-free offline calibration of commercial optical see-through head-mounted displays with simplified procedures," *IEEE Access*, vol. 8, pp. 223661–223674, 2020, doi: 10.1109/ACCESS.2020.3044184.
[24] X. Wang, *Authoritative Guide to AR Development-AR Foudation*. Beijing, China: Tsinghua Univ. Press, 2020, pp. 33–34.
[25] FBX. *Autodesk*. Accessed: Aug. 20, 2024. [Online]. Available: https://www.autodesk.com/products/fbx/overview
[26] N. C. Hashim, N. A. A. Majid, H. Arshad, H. Hashim, and Z. Abdi Alkareem Alyasseri, "Mobile augmented reality based on multimodal inputs for experiential learning," *IEEE Access*, vol. 10, pp. 78953–78969, 2022, doi: 10.1109/ACCESS.2022.3193498.
[27] M. A. Maneli and O. E. Isafiade, "A multifactor comparative assessment of augmented reality frameworks in diverse computing settings," *IEEE Access*, vol. 11, pp. 12474–12486, 2023, doi: 10.1109/ACCESS.2023.3242238.
[28] W. Ye, H. Li, T. Zhang, X. Zhou, H. Bao, and G. Zhang, "SuperPlane: 3D plane detection and description from a single image," in *Proc. IEEE Virtual Reality 3D User Interface (VR)*, Mar. 2021, pp. 207–215, doi: 10.1109/VR50410.2021.00042.
[29] L. Royakkers, D. Snijders, and R. van Est, "The ten commandments for responsible augmented reality," in *New Trends in Disruptive Technologies, Tech Ethics and Artificial Intelligence*. Cham, Switzerland: Springer, 2021, pp. 121–132, doi: 10.1007/978-3-030-87687-6_13.

JINFANG LI received the Ph.D. degree in engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2004. She is currently with the Electromechanical Engineering, Guangdong University of Technology (GDUT). Her research interests include visualization research, virtual medical application, and industrial simulation.

YONGXI LIU received the B.S. degree from the Department of Mechanical and Electrical, Guangdong University of Technology (GDUT), Guangzhou, China, in 2021, where he is currently pursuing the M.S. degree with the Department of Mechanical Engineering. His current research interests include virtual reality, augmented reality, and visualization research.

LIBAO XIAO received the B.S. degree from the Department of Mechanical and Electrical, Guangdong University of Technology (GDUT), Guangzhou, China, in 2022, where he is currently pursuing the M.S. degree with the Department of Mechanical Engineering. His current research interests include virtual medical application, industrial simulation, and visualization research.

JIAHONG CAI received the M.S. degree from the Department of Mechanical Engineering, Guangdong University of Technology (GDUT), Guangzhou, China, in 2021. He is currently with CET, Shenzhen, China. His research interest includes power system simulation.

HANWU HE received the Ph.D. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2001. He is currently the Principal of the Guangdong Polytechnic of Industry and Commerce. He is also the Director of the Virtual Reality and Visualization Engineering Research Center of Guangdong Province. His research interests include virtual reality, augmented reality, and intelligent manufacturing. His research projects have been sponsored by several organizations, including the Natural Science Foundation of China.

XIAOWEI ZHANG received the Ph.D. degree in mechanical design and theory from Northeastern University, China. She is currently a Lecturer with the Guangdong University of Technology. Her research interests include theory and technology of innovative design and simulation of mechanical products.