Empowering End-User in Creating eXtended Reality Content with a Conversational Chatbot

Jacopo Mereu[1 [0009−0008−7521−7876]], Valentino Artizzu[1 [0000−0003−0263−2434]], Alessandro Carcangiu[1], Lucio Davide Spano[1 [0000−0001−7106−0463]], Ludovica Simeoli[2] Andrea Mattioli[2 [0000−0001−6766−7916]], Marco Manca[2 [0000−0003−1029−9934]], Carmen Santoro[2 [0000−0002−0556−7538]], and Fabio Paternò[2 [0000−0001−8355−6909]]

[1] University of Cagliari, Dept. of Mathematics and Computer Science, Via Ospedale 72, 09124, Cagliari, Italy
{jacopo.mereu, valentino.artizzu, alessandro.carcangiu, davide.spano}@unica.it
[2] ISTI-CNR, Human Interfaces in Information Systems (HIIS) Laboratory, Via G. Moruzzi 1, 56124, Pisa, Italy
{ludovica.simeoli, andrea.mattioli, marco.manca, carmen.santoro, fabio.paterno}@isti.cnr.it

Abstract. Recent advancements in eXtended Reality (XR) technologies have found application across diverse domains. However, creating complex interactions within XR environments remains challenging for non-technical users. In this work, we present EUD4XR, a project aiming to: i) empower end-user developers (EUDevs) to customize XR environments by supporting virtual objects and physical devices; ii) involve an intelligent conversational agent which assists the user in defining behaviours. The agent can handle multimodal input, to drive the EUDev during the rule authoring process, using contextual knowledge of the virtual environment and its elements. By integrating conversational assistance, EUD4XR seeks to lower further the usage barriers for end-users to personalize XR experiences according to their needs.

Keywords: extended reality · end-user development · immersive authoring · large language model · context · multimodal input · meta-design · event-condition-action rules

1 Introduction

Over the past decade, we have witnessed a significant maturation process in eXtended reality (XR) devices, exemplified by products such as Meta Quest, Microsoft HoloLens, and Steam Valve Index, along with connected smart devices, and the combination of them. Once primarily associated with entertainment, these technologies have now found applications in diverse domains such as automotive, healthcare, wellness, and manufacturing. As these tools have matured, the research community and commercial/open-source products have prioritized

empowering end-user developers (EUDevs), individuals possessing expertise in specific domain applications but lacking professional developer skills, to personalize or adjust these applications to suit their specific needs.

While existing solutions facilitate the creation of XR environments [17, 36, 27, 34] and define the behaviour for virtual objects [2, 9, 5] and physical devices [23, 24, 4, 14, 8, 19], they do not completely eliminate user errors resulting from potential misunderstanding. A possible solution to address this issue should: i) propose an immersive environment that empowers end users to customize the behavior of virtual and physical objects within XR environments using rules expressed in natural language, and ii) implement an intelligent conversational agent capable of managing multimodal inputs to guide users through the customization process. In this paper, we review the state of the art on this topic and examine the engineering challenges involved in building such a chatbot, including multimodal input processing, intent-to-rule conversion, and available interaction exploration capabilities.

2 Related Work

Modern professional tools like Unity, Unreal, and WebXR allow users to build and define XR environments and their logic. Such tools require advanced programming knowledge, making them unsuitable for end-user developers (EUDevs).

For this reason, EUD authoring tools are designed to enable EUDevs to address these tasks. The ceiling of these tools may not be as high as that of professional tools, but they can still provide powerful support while keeping the entry knowledge barrier low. We can distinguish between non-immersive (desktop-based) and immersive (head-mounted display-based) tools in the literature. The former are well-explored, as they leverage already investigated EUD techniques from screen-based applications, while the latter attempt to adapt classic EUD techniques for the domain of XR, which is less explored and documented. Various commercial and open-source desktop tools exist for XR environment creation. Tools like Hubs [3] and Spoke [4] help non-technical users in creating virtual experiences. Fungus [5] focuses on supporting novice developers to create interactive storytelling by using flowcharts. Immersive tools have the advantage that, since the user is immersed in the augmented environment, they can perform operations more intuitively through direct manipulation. ECARules4All [5] supports end-users in designing XR experiences. The Template Builder (TB) creates virtual environments and tags virtual objects from a built-in taxonomy. The End-User Developer (EUDev) personalizes this pre-built content by defining object behaviours through natural-language rules. These rules follow the event-condition-action (ECA) scheme. Immersive authoring tools also find applications in the automation domain. MagicHand [33] and HoloHome [23] support end-users in

[3] `https://hubs.mozilla.com/`
[4] `https://hubs.mozilla.com/spoke/`
[5] `https://fungusgames.com/`

supervising a restricted set of IoT devices, while BricklAyer [32] and MagiPlay [31] leverage immersive authoring and building blocks metaphor to help, respectively, adults and children without programming skills. However, both tools have limitations in terms of the complexity of interactions they can support. On the other hand, immersive tools are less flexible and require more focus compared to non-immersive tools.
Significant efforts have been made in Language Models in recent years, leading to the development of Large Language Models (LLMs). These models have been utilized as smart conversational agents in various fields, including medicine, judiciary, and education [35, 25, 20].

Researchers are exploring LLMs' capabilities also to support users in creating, modifying, and prototyping XR environments. While early works focused on integrating LLMs within Unity [28, 26], the research community has begun to exploit them in more sophisticated frameworks that automate certain tasks. For instance, in [12], the authors propose a framework that leverages LLMs to assist users in generating virtual objects to populate 3D scenarios. The framework consists of the use of GPT-4, prompt-tuned in several tasks: i) Planner, analyzes the user's prompt and breaks it down into simpler tasks; ii) Scene Analyzer, examines the Unity scene the user is working on, generating a JSON representation of the scene; iii) Skill Library, searches for pre-existing code and elements, such as meshes or textures, to solve each simpler task; iv) Builder, generates the code to resolve the problem starting from the elements identified by the Skill module; v) Inspector, iteratively verifies the generated code received from the Builder. The Builder and Inspector modules may engage in several iterative steps, gradually refining the solution. Once the Inspector approves, the generated code is compiled and executed on the Unity engine to produce the user's desired functionality.

3 EUD4XR in a Nutshell

In the EUD4XR project, we aim to introduce two key functionalities: i) supporting end-users without programming skills to define XR automations, involving both virtual content and physical devices; ii) lower the threshold for creating such automations through an intelligent conversational agent.

Considering that, in general, XR environment are quite complex to define from scratch, we define a workflow similar to the one used in Content Management Systems (CMS) for the web, where users wanting to build a website but lacking development skills start from a template and configure it iteratively adapting the starting point to their needs.

The engineering process we envision is an application of a meta-design model [3, 13, 11] where we define three roles (Figure 1):

- The **Element Builder** (EB), who creates generic virtual templates and annotates virtual objects and physical devices. This role typically corresponds to professional developers, who have the technical knowledge for modelling

the objects and creating the XR system that will be open for end-user customisation.
- the **End-User Developer** (EUDev), who customises these templates and personalizes their behaviours using a rule language. This role typically represents the domain expert, thereby a role that does not have any specific skills in programming, but is interested in customising a joint behaviour of virtual and real objects.
- the **Final User**, who represents the final consumer of the customised XR contents. This role can, in some situations, coincide with the role of End User developer, when the latter one also benefits of the created customisations.

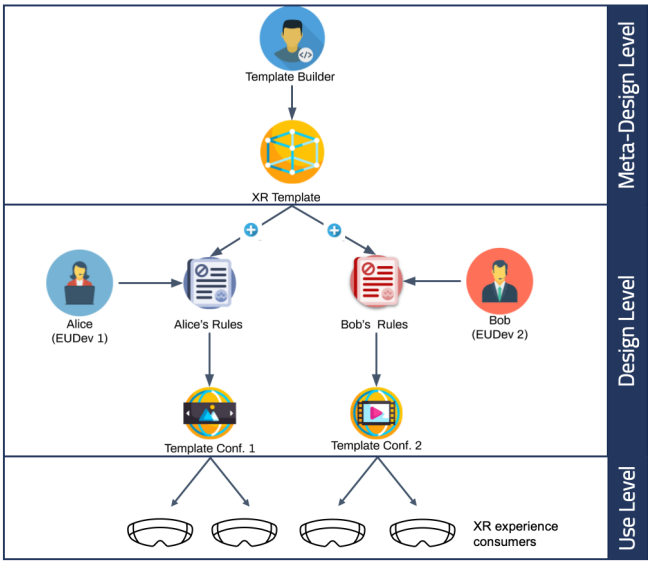

Fig. 1. Roles for the meta-design of XR environments in EUD4XR.

3.1 Object Taxonomy and Rule Language

To support both EBs and EUDevs in managing and defining interactions, our solution requires describing the properties (i.e., the variables defining the state) and the actions (i.e., the supported operations) each virtual and physical object supports through a vocabulary understandable for EUDevs. We started from the extendable taxonomy described in [5] defined for virtual objects, to include physical objects as well. This taxonomy categorizes entities into two types: Objects and Behaviors. The former represents entities based on perceivable characteristics such as shape or functionality, while the latter encompasses actions that

entities can perform. In our approach, each entity, such as a smart light or a virtual chair model, is assigned to an Object category and can possess zero or more Behaviors. Notably, entities in different Object categories can share common Behaviors, enabling flexible customization across diverse entities. For example, consider a smart chandelier item: it belongs to the Electrical Appliance category and cannot be assigned to the Vehicle category. However, both a smart chandelier and a truck emit light, sharing a common Behaviour. The EB annotates the environmental objects with the taxonomy types, while the EUDev exploits tagged objects for defining behaviours.

The EUDev defines object behaviours through a natural language rule system based on an event-condition-action (ECA) schema. EUDevs interact with the rule management system in an immersive way. Each rule follows a "WHEN-IF-THEN" structure, where the WHEN block delineates the trigger event that activates the rule; the IF block is optional and describes the conditions that must be met for the rule's trigger to be raised, and EUD4XR should support one or more conditions; and the THEN block illustrates the action, or actions, to be performed when the rule is triggered.

Existing works adopt the ECA schema to focus on event-type triggers, leveraging states as filter conditions. EUD4XR differentiates, and handles triggers based on their type [18]:

1. Instantaneous action. It does not cause a state change in the environment and is managed through a single rule.
2. Temporary state change. It involves a temporary state change and automatically returns to the starting state after a certain time.
3. Prolonged state change. The object's state does not automatically revert, and EUD4XR may handle it by defining a rule for each possible state.

3.2 AI Rule Authoring

To better support EUDevs in creating XR environments, they may interact with a chatbot. Since the user is immersed, speaking is preferred over writing. Given that the user will interact through their voice, they may find it natural to help themselves by pointing out elements of the XR environment. Additionally, the user may naturally look at some of the fundamental elements.

In summary, our chatbot must be able to manage a multimodal input stream, comprehending user voice and gaze, as well as their pointing, if any. This input will be merged with the environment context, meaning the information about the virtual elements and their status, such as the ECA tag, if any, and by the custom annotations (i.e., metadata on classes, methods and instance variables) specifying the high-level actions each tag can perform. The model should be capable of reading and synchronizing the sources to extrapolate the user's intent in ECA rules. The intent must contain four key pieces: the temporary aspect (when?), the action part (what?), the spatial positioning (where?), and the objects involved (who?). If any key piece is not provided by the user, the chatbot should explicitly ask for it by directing to the user further questions either for

specifying missing parts in the concerned rule, or for removing possible ambiguities existing in its current specification. If a user's intent is not feasible, the chatbot should explicitly state this to users, suggesting the closest feasible intention could be. For example, if they say "When the user selects the door [points to the door they are referring to], then the door should fly over that shelf [points to the shelf]", and knowing that a door has no high-level action for flying, the chatbot should reply with something like "The door cannot perform the action *flying*. Instead, you can use the following actions: close, open, and lock".

To understand how end-users define automations when interacting with an intelligent agent, we have conducted two studies. The two studies have been both set up as "Wizard of Oz" and have explored two different domains to be more general: a smart home, and a virtual museum scenario. The results of the user studies will be used to tune the prompt of a model so that it can more easily recognise users' automation intentions. The studies both analysed not only how users express themselves when specifying the creation of new rules in specific scenarios, but also how they define their customisation intentions when they want just to modify existing rules. An additional goal is to discover a possible interaction paradigm or interaction phases during the dialogue between the agent and the users so that we can design the agent in guiding end users in the automation definition.

4 Engineering Challenges

From an engineering point of view, defining an intelligent agent that can support a user without programming skills in creating automation presents different challenges. LLMs' rapid scale and spread have provided the scientific community and businesses with a powerful tool for processing several data types, quickly creating virtual assistants for repetitive tasks, and enhancing customer care. One factor contributing to the rapid adoption of LLMs is their ability to extract relevant information to answer questions across a wide range of topics, resulting from the training on large text corpora.

Literature explores several solutions that do not rely on LLMs [7, 6, 16]. However, these virtual assistants may limit users' ability to express themselves freely, as these agents need a rigid dialogue flow. In contrast, LLMs have a superior capacity to understand semantics independently of the exact words the user chooses.

A direct application of LLMs to domain-specific problems still requires enhancing them through custom knowledge bases [21] since the quality of LLMs' answers decreases when the dialogue focuses on topics for which they have not been trained [15]. We can envisage two opposite solutions in defining an LLM: fine-tuning a new model or prompt-tuning a pre-trained model. Each solution presents pros and cons; for the first solution, the fine-tuning of a local model can guarantee a faster execution and no privacy issues and fees; however, there is a need for expensive hardware to train the model and create a lengthy dataset can be challenging. On the other hand, the prompt-tuning approach exploits previous

knowledge of the LLM and the In-Context Learning method, where the model learns to address a new task during inference by receiving a prompt, including task examples. However, a possible limitation of prompt-tuning a generic model is that, since the model was pre-trained on a wide range of data, it is not necessarily optimized for a specific domain. We are moving toward a prompt-tuning solution because it proved to be less expensive in terms of the resources needed to define a large dataset and to train the model accordingly. From an interactive system engineering point of view, it is very common to require LLM-based agents that should provide good responses to domain-specific questions, considering that seldom the goal is to provide a general-purpose answering agent. This is also the case for EUD4XR, where we need LMM support to create a chatbot that supports automation creation in an XR environment. A generic LLM can interpret the user's prompt to extract the overall concept of the automation they wish to create by leveraging its generic knowledge. To overcome this problem and improve response accuracy, additional context can be provided to the model along with the user prompt. This context consists of data that allows the model to generate a more appropriate and relevant response, improving the LLM's ability to adapt to the user's query without the need for a huge number of training examples [10].

In the following subsections, we explore and analyse what we consider to be the main challenges in developing an intelligent agent that can assist users without programming skills in creating interactions and behaviours within specific scenarios, such as home automation or the customisation of a VR museum application.

4.1 Multimodal Input

XR interfaces can exploit the surrounding space to provide context for the information. This might encourage end-users to take advantage of different modalities to refer to physical and virtual objects, e.g., by pointing, using demonstrative pronouns, etc. The intelligent agent should have access to multiple data sources to support the user in generating appropriate automation for the environment. The research community is actively studying and developing LLMs able to handle different data sources beyond text, including video, audio or images [22, 30, 29]. However, currently, no model can integrate all the functionalities the solution we propose needs, e.g., gesture pointing and spatial information. The challenge here is to collect the information from different input streams and interpret and transform it into something the LLM can consider for generating its answers. For instance, the relevant contextual information may be a list of available physical and virtual objects, their executable functionalities, the existing rules, etc.

Figure 2 shows an example of the rule authoring we envision in an XR environment. Consider a scenario where the manager of a cleaning company needs to train new employees on room sanitization protocols, such as Covid-19 disinfection procedures. To ensure effective training, the manager exploits an XR environment where new staff can interact with a blend of virtual and physical

elements. Using the natural language, the manager creates an automation that makes the training interactive.

For instance, the automation defined by the manager could be expressed as follows: "*When the employee enters this room, virtually highlight these surfaces to disinfect*". The manager's statement includes references to the services virtual devices provide (highlight), demonstrative adjectives specifying which elements the user is referring to (surfaces), rule triggers (When the employee enters this room), condition (to disinfect), and ambiguous parameters whose definition need further questions by the chatbot (these surfaces). The LLM agent requires data about the current state of the XR environment (e.g., the framed device or the pointed virtual object), a description of the API supported by both physical and virtual devices to control them, a system prompt describing what the agent has to produce (an interaction rule) and, of course, the user's utterance and the context from previous interactions.

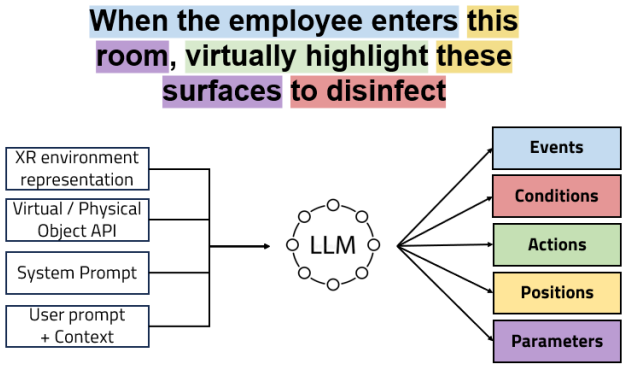

Fig. 2. Example of authoring an interaction rule using multimodal inputs.

4.2 Guidance from intent to a correct rule

The intelligent agent guides the end user from formulating intent to generating valid automation within the XR environment. Starting from a similar problem addressed by Nielsen and Norman Group [1], where the authors describe the steps users follow to generate images with AI, we revisited the process of generating valid automation with the support of an intelligent agent according to these four stages: Define, Explore, Refine and Confirm.

In the **Define** stage, users declare the automation objectives they want to reach, often using high-level, ambiguous terms. This could imply specifying the final outcome that users want to achieve, or the automation "intent", i.e., "*I don't want to feel cold when I return home*" or "*I want the virtual door to become invisible*". If users can express their automation in terms of specific sensors and

devices from the beginning, this phase can be skipped and users can go directly to the next one (Explore).

Since the automation's objectives identified at the end of the Define phase can be realised in different manners, in the **Explore** stage, the user explores and determines which objects and capabilities will be involved to realise the desired automation concretely. Thus, while the previous phase answers to "what" should be achieved, this phase aims to understand "how" to realise the intent defined before. Here, the intelligent agent, leveraging the description of the environment and its content, supports the user in identifying the objects, virtual or physical, to utilise to achieve the intention specified in the previous stage. For instance, it may suggest using the heat pump to warm the house. The output of this stage is still an incomplete automation, where the user has typically selected the objects and their capabilities that will form the basis of the automation.

In the **Refine** stage, the chatbot and the user refine the automation to finally meet the desired goal. In this stage, the agent may ask the user to specify any necessary parameters, such as the time or the minimum temperature threshold for activating the heat pump or the distance from home, which may trigger the event "*when I return home*", exploiting the smartphone GPS to understand when the user returns home. It may also request to disambiguate a "reverse" scenario, such as what to do when the house temperature becomes too high, or suggest adding conditions, i.e., to switch off the heat pump once a certain temperature is reached. The output of this step is a valid automation within the environment.

Finally, the **Confirm** stage is where the end-user confirms and saves the automation generated with the help of the intelligent agent, or chooses to discard it.

The phases described in [1] originally followed a linear sequence. However, since the process of generating valid automations with the aid of a chatbot is iterative and flexible, we envisage that, when exploiting such steps for automation generation, some of such phases may be re-visited several times, or even skipped, before arriving at the final version of the automation.

4.3 Exploring the Available Interactions

Another feature we want to support through the conversational agent is exploring the services offered by different physical devices and the actions supported by virtual content. This will help end-users find possible automation suggestions that could be useful for them. In addition, the interface must support the user in recalling or finding out which are the currently defined automations, i.e., rules, associated to a given object or defined in the environment. The list of available interactions provides the agent with an additional source of information for building knowledge about the objects present in the system and their capabilities. This data can be made available to the agent in different ways, for example, through an external service or by injecting it into the context. Regardless of the chosen approach, the support system should consistently be able to:

- Access the list of all automations defined up to that moment.
- Retrieve all the automation's components, i.e., triggers, conditions, and actions, as well as their status if provided by the automation management system.
- Convert the list of automations and their respective components into a textual format.

5 Conclusion

In this work, we analysed existing work on the EUD authoring tools for creating XR environments and interactions, also outlining a solution for an immersive environment that empowers end-users to customize the behaviour of virtual and physical objects in XR settings using rules expressed in natural language. The solution also includes the support of an intelligent conversational agent which leverages the use of LLMs to lower the barrier to entry for non-technical users, enabling them to create personalized and immersive XR experiences. Future work will be dedicated to implement and evaluate the effectiveness and usability of our solution in real-world scenarios.

6 Acknowledgements

This work has been supported by the Italian PRIN 2022 "EUD4XR: End-User Development for eXtended Reality" funded by the Italian MUR and European Union - NextGenerationEU under grant PRIN 2022 EUD4XR (Grant F53D23004380006) `https://prin.unica.it/eud4xr/`.

References

1. The 4 Stages of AI Image Generation: An Experience Map, `https://www.nngroup.com/articles/ai-imagegen-stages/`
2. Adão, T., Pádua, L., Fonseca, M., Agrellos, L., Sousa, J.J., Magalhães, L., Peres, E.: A rapid prototyping tool to produce 360 video-based immersive experiences enhanced with virtual/multimedia elements. Procedia computer science **138**, 441–453 (2018)
3. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R., Piccinno, A., Zhu, L.: On the transferability of a meta-design model supporting end-user development. Universal Access in the Information Society **14**, 169–186 (2015)
4. Ariano, R., Manca, M., Paternò, F., Santoro, C.: Smartphone-based augmented reality for end-user creation of home automations. Behaviour & Information Technology **42**(1), 124–140 (2023)
5. Artizzu, V., Cherchi, G., Fara, D., Frau, V., Macis, R., Pitzalis, L., Tola, A., Blecic, I., Spano, L.D.: Defining configurable virtual reality templates for end users. Proceedings of the ACM on Human-Computer Interaction **6**(EICS), 1–35 (2022)
6. Asunis, L., Frau, V., Macis, R., Pireddu, C., Spano, L.D.: Pac-bot: Writing text messages for developing point-and-click games. In: International Symposium on End User Development. pp. 213–221. Springer (2021)

7. Barricelli, B.R., Casiraghi, E., Valtolina, S.: Virtual assistants for end-user development in the internet of things. In: International Symposium on End User Development. pp. 209–216. Springer (2019)
8. Barricelli, B.R., Valtolina, S.: A visual language and interactive system for end-user development of internet of things ecosystems. Journal of Visual Languages & Computing **40**, 1–19 (2017)
9. Blečić, I., Cuccu, S., Fanni, F.A., Frau, V., Macis, R., Saiu, V., Senis, M., Spano, L.D., Tola, A.: First-person cinematographic videogames: Game model, authoring environment, and potential for creating affection for places. Journal on Computing and Cultural Heritage (JOCCH) **14**(2), 1–29 (2021)
10. Chamieh, I., Zesch, T., Giebermann, K.: Llms in short answer scoring: Limitations and promise of zero-shot and few-shot approaches. In: Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024). pp. 309–315 (2024)
11. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: A meta-design approach to end-user development. In: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05). pp. 308–310. IEEE (2005)
12. De La Torre, F., Fang, C.M., Huang, H., Banburski-Fahey, A., Amores Fernandez, J., Lanier, J.: Llmr: Real-time prompting of interactive worlds using large language models. In: Proceedings of the CHI Conference on Human Factors in Computing Systems. pp. 1–22 (2024)
13. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-design: a manifesto for end-user development. Communications of the ACM **47**(9), 33–37 (2004)
14. Fogli, D., Peroni, M., Stefini, C.: Imathome: Making trigger-action programming easy and fun. Journal of Visual Languages & Computing **42**, 60–75 (2017)
15. Galitsky, B., Chernyavskiy, A., Ilvovsky, D.: Truth-o-meter: Handling multiple inconsistent sources repairing llm hallucinations. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2817–2821 (2024)
16. Gallo, S., Paterno, F.: A conversational agent for creating flexible daily automation. In: Proceedings of the 2022 International Conference on Advanced Visual Interfaces. pp. 1–8 (2022)
17. Garzotto, F., Gelsomini, M., Matarazzo, V., Messina, N., Occhiuto, D.: Xoom: An end-user development tool for web-based wearable immersive virtual tours. In: Web Engineering: 17th International Conference, ICWE 2017, Rome, Italy, June 5-8, 2017, Proceedings 17. pp. 507–519. Springer (2017)
18. Huang, J., Cakmak, M.: Supporting mental model accuracy in trigger-action programming. In: Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing. pp. 215–225 (2015)
19. Ikeda, B., Szafir, D.: Programar: Augmented reality end-user robot programming. ACM Transactions on Human-Robot Interaction (2024)
20. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al.: Chatgpt for good? on opportunities and challenges of large language models for education. Learning and individual differences **103**, 102274 (2023)
21. Kumar, P.: Large language models (llms): survey, technical frameworks, and future challenges. Artificial Intelligence Review **57**(9), 260 (2024)
22. Lyu, C., Wu, M., Wang, L., Huang, X., Liu, B., Du, Z., Shi, S., Tu, Z.: Macaw-llm: Multi-modal language modeling with image, audio, video, and text integration. arXiv preprint arXiv:2306.09093 (2023)

23. Mahroo, A., Greci, L., Sacco, M.: Holohome: an augmented reality framework to manage the smart home. In: Augmented Reality, Virtual Reality, and Computer Graphics: 6th International Conference, AVR 2019, Santa Maria al Bagno, Italy, June 24–27, 2019, Proceedings, Part II 6. pp. 137–145. Springer (2019)
24. Manca, M., Paternò, F., Santoro, C., Corcella, L.: Supporting end-user debugging of trigger-action rules for iot applications. International Journal of Human-Computer Studies **123**, 56–69 (2019)
25. Martin, L., Whitehouse, N., Yiu, S., Catterson, L., Perera, R.: Better call gpt, comparing large language models against lawyers (2024)
26. Matukumalli, V., Naga Sasidhar Maddi, S., Krishna Angirekula, K., Reddy Pulicherla, V., Senthil kumar, A., Maridurai, T., Sathish, T., Kasinathan, D.: Augment reality chatbot using cloud. Materials Today: Proceedings **46**, 4254–4257 (2021). https://doi.org/https://doi.org/10.1016/j.matpr.2021.03.058, `https://www.sciencedirect.com/science/article/pii/S2214785321020721`, international Conference on Materials, Manufacturing and Mechanical Engineering for Sustainable Developments-2020 (ICMSD 2020)
27. Menestrina, Z., De Angeli, A.: End-user development for serious games. New Perspectives in End-User Development pp. 359–383 (2017)
28. Morales, P., Showalter-Bucher, R.A.: Towards large language models at the edge on mobile, augmented reality, and virtual reality devices with unity. In: System and Architecture for Generative AI on the Edge/Mobile Platforms (SAGE) (2023), `https://openreview.net/forum?id=ahVsd1hy2W`
29. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
30. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision. In: International conference on machine learning. pp. 28492–28518. PMLR (2023)
31. Stefanidi, E., Arampatzis, D., Leonidis, A., Korozi, M., Antona, M., Papagiannakis, G.: Magiplay: An augmented reality serious game allowing children to program intelligent environments. Transactions on Computational Science XXXVII: Special Issue on Computer Graphics pp. 144–169 (2020)
32. Stefanidi, E., Arampatzis, D., Leonidis, A., Papagiannakis, G.: Bricklayer: a platform for building rules for ami environments in ar. In: Advances in Computer Graphics: 36th Computer Graphics International Conference, CGI 2019, Calgary, AB, Canada, June 17–20, 2019, Proceedings 36. pp. 417–423. Springer (2019)
33. Sun, Y., Armengol-Urpi, A., Kantareddy, S.N.R., Siegel, J., Sarma, S.: Magichand: Interact with iot devices in augmented reality environment. In: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). pp. 1738–1743. IEEE (2019)
34. Takala, T.M.: Ruis: A toolkit for developing virtual reality applications with spatial interaction. In: Proceedings of the 2nd ACM symposium on Spatial user interaction. pp. 94–103 (2014)
35. Thirunavukarasu, A.J., Ting, D.S.J., Elangovan, K., Gutierrez, L., Tan, T.F., Ting, D.S.W.: Large language models in medicine. Nature medicine **29**(8), 1930–1940 (2023)
36. Yigitbas, E., Klauke, J., Gottschalk, S., Engels, G.: Vreud-an end-user development tool to simplify the creation of interactive vr scenes. In: 2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 1–10. IEEE (2021)