MechARspace: An Authoring System Enabling Bidirectional Binding of Augmented Reality with Toys in Real-time

Zhengzhe Zhu*
zhu714@purdue.edu
School of Electrical & Computer Engineering, Purdue University
West Lafayette, IN, USA

Ziyi Liu*
liu1362@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Tianyi Wang
wang3259@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Youyou Zhang
zhan3264@purdue.edu
School of Electrical & Computer Engineering, Purdue University
West Lafayette, IN, USA

Xun Qian
qian85@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Pashin Raja
praja@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Ana Villanueva
villana@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Karthik Ramani
ramani@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

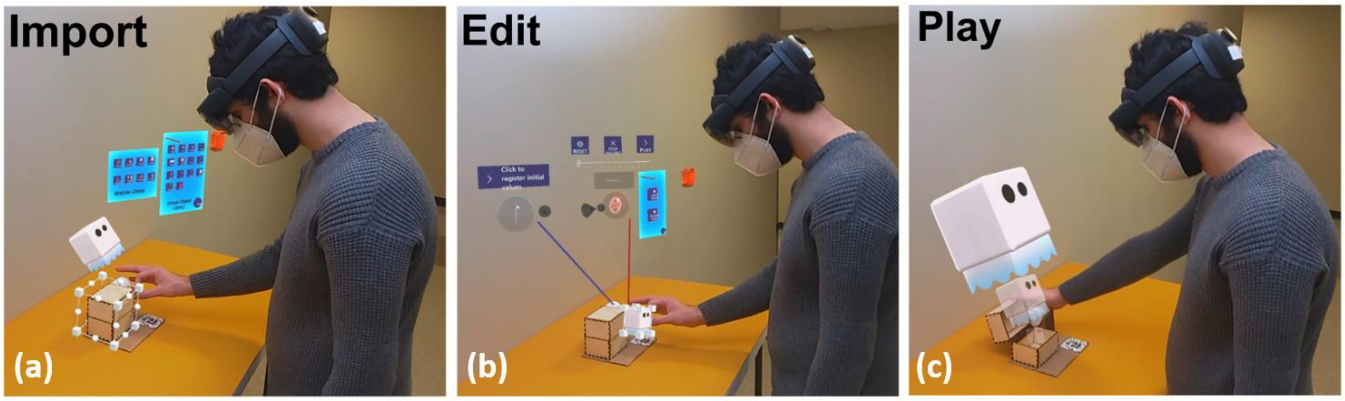

Figure 1: An overview of MechARspace workflow. (a) A user starts with a chest (physical toy) which contains a rotation module (part of our IoT toolkit). The chest, with its module, is imported into the AR scene by the user defining a bounding box around it. (b) The user authors the interaction between the virtual ghost and the rotation module of the physical chest through demonstration and input-output visual programming. The ghost pops out of the chest with respect to the angle the user sets for the rotation module. (c) The user plays with his authored AR-enhanced toy.

ABSTRACT

Augmented Reality (AR), which blends physical and virtual worlds, presents the possibility of enhancing traditional toy design. By leveraging bidirectional virtual-physical interactions between humans and the designed artifact, such AR-enhanced toys can provide more playful and interactive experiences for traditional toys. However, designers are constrained by the complexity and technical difficulties of the current AR content creation processes. We propose MechARspace, an immersive authoring system that supports users to create toy-AR interactions through direct manipulation and visual programming. Based on the elicitation study, we propose a bidirectional interaction model which maps both ways: from the toy inputs to reactions of AR content, and also from the AR content to the toy reactions. This model guides the design of our system which includes a plug-and-play hardware toolkit and an in-situ authoring

*Both authors contributed equally to this research.

UIST '22, October 29-November 2, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10.
https://doi.org/10.1145/3526113.3545668

interface. We present multiple use cases enabled by MechARspace to validate this interaction model. Finally, we evaluate our system with a two-session user study where users first recreated a set of predefined toy-AR interactions and then implemented their own AR-enhanced toy designs.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**.

KEYWORDS

Augmented Reality, immersive authoring, program by demonstration, modular IoT toolkit

ACM Reference Format:
Zhengzhe Zhu, Ziyi Liu, Tianyi Wang, Youyou Zhang, Xun Qian, Pashin Raja, Ana Villanueva, and Karthik Ramani. 2022. MechARspace: An Authoring System Enabling Bidirectional Binding of Augmented Reality with Toys in Real-time. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3526113.3545668

1 INTRODUCTION

Augmented Reality (AR) leverages forms and features of both physical and virtual environments. AR thus provides designers with the potential to enhance physical toys with visualizations and interactions beyond physical constraints. A handful of AR-enhanced toys have recently emerged, from commercial products like Mario Kart Live [7] and Lego Hidden Sides [6] to research projects like Project Zanzibar [80] and Checkmate [36]. However, existing tools for creating AR applications for these toys are dedicated for the use by experienced professionals and have high technical barriers to entry [11, 12, 61]. Thus, novice designers miss out on the opportunity of imagining, designing, and implementing AR-enhanced toys based on their unique ideas. To address this gap, we develop an authoring system which empowers end-users to quickly prototype AR applications to personalize their physical toys.

The first step towards developing the authoring system is to understand the nature of toy-AR interactions. We summarize that these interactions take place bidirectionally between physical and virtual worlds: (a) physical toys as triggers to actuate AR content, (b) AR content as triggers to actuate physical toys. As an example of (a), in LEGO Hidden Side [6], users can manipulate the LEGO blocks to unlock a hidden storyline displayed using AR. As for (b), in the racing car game, Mario Kart Live: Home Circuit [7], users can throw a virtual bomb on opponents' physical cars to stop them as if the bomb were real. In (a), the toys serve as tangible user interfaces (TUI) giving users access to the virtual world. In (b), the reactions of the toys to AR animations make the interaction more realistic and immersive [74]. To the best of our knowledge, a unified design framework for bidirectional toy-AR interactions has not been established in prior works.

Conventional AR authoring tools (e.g., Unity3D [11], Unreal [12], ARCore [1], ARKit [2], etc.) decouple the programming and testing environment [61, 76, 81]. Developers have to project the status of the 3D physical toy from different perspectives while programming AR animations, which requires expertise and is time-consuming [59, 91]. In contrast, the immersive experience supported by AR-HMD fosters the evolution of authoring workflows in an in-situ approach [81]. By blending the authoring process into the AR environment, users have contextual information—the status viewed from different perspectives—of the physical toy and its associated AR content. For example, let us imagine a scenario in which a user wants to program a physical toy such as a *Transformer* robot, which can trigger different special effects (e.g., fire, laser beam) along with corresponding fighting poses (e.g., arms forming an X or a T) using AR. The user can simply change the robot to fighting poses and program the associated AR animations/effects on the spot, while referring to the robot's pose for spatial context. In addition, the WYSIWYG (What You See Is What You Get) metaphor [21] enables users to create animations through direct manipulation within AR without writing lines of code, which lower the entry barrier of AR authoring [74, 81].

Similar to authoring AR content, authoring physical behaviors of toys should be equally intuitive and accessible to end-users. To lower the barriers of physical programming, researchers have designed various robotics modules can record the user's demonstration and play it back to animate the robot [66, 66, 71, 75]. With such tools, users can design and create without any low-level programming knowledge. Inspired by these works, we develop a toolkit that allows users to embed virtual effects to the physical structures of toys, and then create an interactive virtual-physical experience through programming by demonstration.

We propose MechARspace, an in-situ authoring system that supports the real-time creation of AR applications to make toys more playful. We explore the design space of physical-virtual interactions between AR and toys and use it to guide our system design. The system includes two parts: a plug-and-play toolkit designed to be embedded into common toys, and an authoring interface built on a pair of optical see-through AR glasses (i.e., Hololens 2 [3]). The toolkit encapsulates different sensors and actuators so that users can effortlessly upgrade toys with AR capabilities without going through lengthy electronic prototyping processes. This toolkit is IoT-enabled, which means it will be constantly connected to the AR headset once imported to the scene. MechARspace facilitates intuitive authoring of both AR animations and toys' actions through direct manipulation, while referring to the physical toy for a contextual reference. Through a trigger-action visual programming interface, users can create bidirectional interaction by utilizing their inputs on toys to trigger AR animations and vice versa. Furthermore, with the real-time sensing and actuating capability of the toolkit, users can instantly play with the authored toys which fosters rapid design exploration.

We propose the following contributions:

- A unified framework of the bidirectional physical-virtual interaction model for the AR-enhanced toys.
- A plug-and-play IoT-enabled modular toolkit with sensing and actuating capability that enables ordinary toys to interact with AR.
- An in-situ and easy-to-use authoring interface for creating toy-based AR applications through demonstration and visual programming.

- A wide range of example toys with their AR applications to validate the applicability of our interaction model and the extensive reach of our authoring system.

2 RELATED WORK

2.1 Bidirectional Interactions in AR-enhanced Toys

A toy is defined as a tangible that is engaging to play with [55]. Recently, researchers have discovered that combining emerging AR technology with traditional toys can significantly expand the interaction space and increase the playfulness [40, 41, 77]. AR generates a composite view for the user, which provides the capability for toys to fuse the matters from virtual and physical worlds [42, 60, 77]. In general, we classify the interactions between AR and toys into two directions.

Physical toys as triggers to AR content: Interactions in this category are inspired by the concept of tangible user interfaces (TUIs) [48, 49] which translates the manipulation of physical objects (as triggers) into animations of AR content. For example, static or dynamic virtual models are anchored onto tracked toys to follow their movement [21, 38]. Apart from this location-oriented binding, AR can be controlled by tangibles through logical connections. RobotQuest [57] is a mixed reality gaming platform where players can control a physical robot to repel virtual enemies projected on the floor. Zhou et al. [92] proposed an AR storytelling tool with which users unlock storylines by manipulating a cube. Project Zanzibar [80] proposed an interaction scenario where users trigger different AR animations by moving and rotating toy figures in the scene.

AR content as triggers to actuate physical toys: Toys nowadays are often equipped with actuators (e.g., servos, motors, LEDs) [28, 43] which enable them to actively react to corresponding AR content. Prior works have exploited AR to create in-stu user interfaces for programming both industrial [50] and recreational robots [26, 37, 85]. Similarly, exTouch [51] utilized an AR-mediated mobile interface to translate users' touchscreen interactions into physical objects' actuations. Meanwhile, researchers have sought to reinforce the sensation that the virtual and real elements coexist by allowing AR to actuate the physical world [23, 74]. For example, MotionBeam [84] proposed an interaction scenario where a cartoon character projected onto the wall can jolt a picture frame out of place. ColabAR [79] introduced a haptic module which will vibrate whenever it collides with a virtual object. In Kobito [18], imaginary agents can move real objects (e.g., a tea caddy) on the table to make themselves more realistic.

A more comprehensive summary of bidirectional physical-virtual interactions could be found in the recent AR and robotics survey paper from Suzuki et al. [74].

Interactions in the previously mentioned systems are predetermined for specific applications by professional developers. The lack of novice-friendly development tools impedes novice designers' ability to integrate personalized AR experiences to their toys. Previous AR authoring systems have allowed users to dynamically superimpose virtual content on top of physical items based on their preferences [32, 53, 78]. However, these authoring systems only enable one-direction of interaction authoring (i.e., AR content is simply superimposed on the physical objects). To capture a more extensive design space, we propose MechARspace, an end-user authoring system that facilitates the bidirectional virtual-physical authoring space of toy-AR interactions.

2.2 Immersive AR Authoring Tools

Development tools for creating AR applications have long been the focus within the HCI community [61, 67, 68, 70, 78]. Traditional desktop-based AR development tools such as Unity [11] and Unreal [12] decouple the programming and testing process. Alternatively, immersive authoring has been proposed to directly blend the authoring process into the AR interaction space [58, 83, 90]. This approach allows for intuitive 3D manipulation instead of 2D programming [67], and accelerates the build-test cycle by enabling on-the-spot evaluation of authored artifact [59]. Following this paradigm, Window Shaping [46] and SceneCtrl [89] empowered the creation of static 3D models and virtual scenes by leveraging the spatial perception of AR devices. ARAnimator [87] and PuppetPhone [16] allowed users to author an animation sequence of a virtual character using a smartphone as the motion controller.

However, AR applications created by these tools are not interactive, and thus, cannot respond to users' actions or real-world phenomena. Researchers have sought to incorporate different sources of input into the authoring workflow to drive dynamic AR animations. Ng et al. [69] facilitated situated AR games development with users' real-time location as trigger. SpatialProto [65] achieved prototyping of interactive AR through capturing human motion for associating with animated virtual content. GestureAR [81] presented an authoring system which supports end-users to define customized gestures for freehand AR applications. ProGesAR [86] allowed users to quickly design proxemic and gestural interactions with real-world IoT by prototyping them in AR.

In particular, several authoring systems have utilized the status of tangible objects as input. iaTAR [58] enabled users to map fiducial markers to common UI elements (e.g. buttons, sliders) to design a tangible AR interface. RealitySketch [76] allowed users to make sketched graphics responsive by binding them with moving objects. With these tools, users freely drew mappings from real-world tangible objects to AR content.

In MechARspace, we take a step further by enabling the authoring of *bi-directional* interactions — MechARspace does not only augment *passive* physical objects, but also incorporates with *actuated* physical objects by leveraging IoT modules, so that users can quickly author an interactive and bi-directional experience for AR-enhanced tangible objects. With immersive authoring, we allow users to constantly refer to their toys for more context information and to test their ideas immediately. Using this strategy, users can quickly validate and iterate their ideas.

2.3 Programming by Demonstration

Programming by Demonstration (PbD) [56, 64] is a technique that allows users to define interactivity by performing examples of the intended behaviors. It can greatly enhance end-user development by overcoming low-level programming details [61, 63]. It was first introduced to the robotics industry [20, 24] where operators demonstrate movements for robots to repeat. Thanks to the embedded

sensors with kinetic memory, this process could be done intuitively by direct manipulation [30]. Similarly, Reactile [75] proposed a tabletop TUI customization approach by manually arranging the objects to desired states.

Meanwhile, PbD also finds its applications in the virtual space. Arora et al. [19] allowed users to author AR/VR animations by holding virtual objects through designated trajectories. AdapTutAR [45] captured body gestures over a given period to generate AR tutorials. Recently, Wang et al. [81] combined PbD with trigger-action visual programming, where the demonstrated hand gestures and AR animations are linked as input and output respectively.

Inspired by prior work, MechARspace applies the metaphor of programming by demonstration to tangible and virtual objects alike. Users' interactions with AR content and actions upon tangible objects are both recorded and later connected together with trigger-action visual programming. Utilizing this strategy, users can define the bidirectional interactivity between input and output elements. Furthermore, our immersive authoring environment can supplement PbD with rich and in-situ visualization of the demonstrated outcome, which brings contextual awareness to the workflow [25].

2.4 Modular IoT Toolkit

Developing physical devices and interfacing them to conventional programming languages has always been difficult for end-users [34, 66, 71]. To address this gap, Phidgets [34] pioneered the approach of packaging input and output devices into modules to hide implementation and construction details from users. Similarly, Physikit [44] introduced an easily configurable toolkit that can be embedded to everyday objects, allowing users to physically visualize IoT data in their homes. To further lower the entry barriers, Topobo [71] and MorphIO [66] adopted the concept of programming by demonstration for their modular toolkit.

Recently, researchers have sought to utilize modular IoT toolkit to enrich AR experiences [54]. The most common application scenario is to utilize capacitive sensing [35, 88], or RFID modules [80] for tracking tangible objects. Compared to traditional vision-based tracking, they are less susceptible for low light or occlusion, thus can provide an uninterrupted experience [73, 80]. Similarly, IMUs have been utilized to detect the gesture input for manipulating AR content [39] and calculating the toy's absolute position [27]. Apart from exploiting toolkit's sensing capability, Cao et al. [26] and Glenn et al. [33] presented modular IoT that could be dynamically actuated by AR animations.

Inspired by these works, we introduce a collection of modular components for our toolkit that could help record human input as well as actively react to dynamic AR animations. Our setup is designed to resonate with the the bidirectional toy-AR interaction model. The modules are intended to provide a plug-and-play experience and help toy designers quickly prototype diverse AR experiences based on traditional toys .

3 INTERACTION FRAMEWORK FOR AR AND TOYS

The interactions between AR and toys are currently scattered across independent systems, creating somewhat a fragmented research landscape. To address this problem, we conducted an elicitation study which helps us combine the affordances of AR and toys to propose a unified framework that summarize these bidirectional interactions.

3.1 Input and Output Model

The input-output model has been widely adopted by previous interaction authoring systems including GesturAR [81], Trigger-Action-Circuits [17], and Kitty [52]. In this model, an interaction involves two components, an input that is initiated by a subject, and an output that is generated by an object in response to the input. The toy-AR joint interaction adopts a similar pattern where the input is the toy's action and the output is the behavior of virtual content, or the other way around. More specifically, using a physical toy as input refers to the process where a user directly moves or manipulates the toy into a certain status and triggers an output. For instance, moving a *Transformer* toy robot from place A to B. Meanwhile, the output refers to the process in which the objects project certain visualizations. For example, sparks coming out from the *Transformer*'s legs.

To categorize input and output interactions between AR and toys, we analyzed DIY-ed toys made by undergraduate students from a senior level toy design class. This class has been taught for about 20 years as an innovative approach for teaching computer-aided design (CAD) and prototyping to mechanical engineering students. At the end of each semester, students (in groups of 4-5) must present their prototype implementations, which usually involve the design of both hardware and software. These presentations were recorded, which helped us better understand the complexity and capabilities of each toy during our analysis. Two co-authors of this paper have been serving as teaching assistants of this class for the past two years, so they have first-hand knowledge of students' toy design processes. We examined a total of 78 toys made by students in the last three semesters (Figure 2). Based on the taxonomy of toys developed by Kudrovitz et al. [55], we classified these toys into the following groups: *Fantasy* (38), *Construction* (18), *Sensory* (11), and *Challenge* (7). We analyzed common features in the toys, in terms of fabrication (hardware parts) and programming (logic and software). Likewise, we summarized the purpose of each toy and envisioned how they could be AR-enhanced within the context meant by the designers. Based on this elicitation process, we compiled a detailed classification of input (Figure 3) and output (Figure 4) interactions across AR (virtual content) and toys (physical content). Each cell in these figures depicts a single example interaction, but represents an entire class of opportunities.

Figure 2: Representative toys designed in the toy design class.

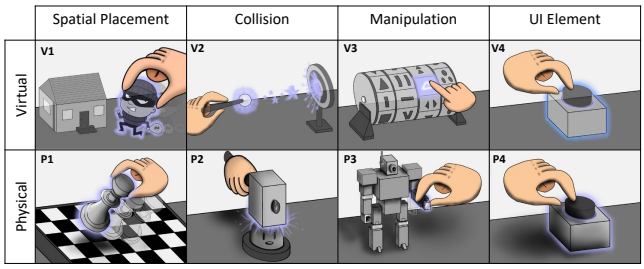

Figure 3: Input taxonomy.

Input 1: *Spatial placement.* Spatial placement relates to the input class in which the user moves an object (physical or virtual) from one place to another. This type of input is common in board games where toy pieces' and figures' movements are meaningful to the game's progress. For instance, in StoryMakAR [33], users place the virtual character on different places to unlock respective storylines.

Input 2: *Collision.* Collision is a special case of spatial placement in which the user holds an object (virtual or physical) to hit another one directly. For example, in the popular toy "Wack-A-Mole", the user wields a physical hammer to smash a physical mole. Similarly, virtual content can also be made to collide with a physical (or virtual) object. For instance, in MotionBeam [84], the user controls the virtual character to hit the picture hanging on the wall.

Input 3: *Manipulation.* Some toys allow users to change their internal status or layout without moving it spatially. For instance, users can move the arms of a *Transformer* toy robot to form different poses. Meanwhile, virtual items with intricate structures could also be manipulated (e.g., a virtual lock to be set at combination 1-2-3-4 to close a chest).

Input 4: *UI element.* Due to legacy bias [22], traditional UI elements such as buttons and knobs are still favored by the users for giving input to the toy. The digital twins of these UI elements are also prevalent in virtual space.

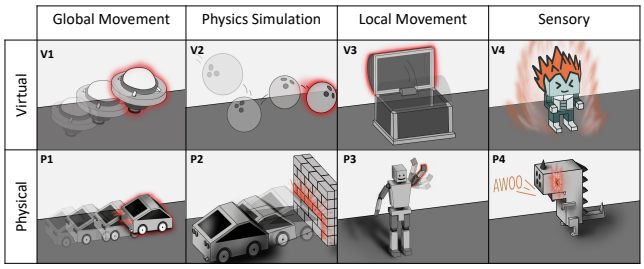

Figure 4: Output taxonomy.

Output 1: *Global movement.* Most car-like toys support this output as they are designed to move at relatively long distances from the user. Similarly, virtual objects can travel around space along any trajectory predefined by the user [19, 81].

Output 2: *Physics simulation.* This type of output is a special addition to the global movement. The difference is that the trajectory of the movement follows the laws of physics and cannot be explicitly defined by the user. For example, a virtual ball would bounce up and down on the physical desk and a physical toy car would bounce back after hitting a virtual wall.

Output 3: *Local movement.* As opposed to the global movement, local movement refers to parts of the toy moving relative to its main body. For instance, the top of the chest opens, or the arm of the robot waves. Depending on the user's preference, the part that moves locally can be virtual or physical [81].

Output 4: *Sensory effect.* We adopt the word "sensory" from Kudrowitz et al.'s paper on toy taxonomy [55], which defines sensory play to involve intentional entertaining of the senses such as hearing, vision, and touch. For instance, the haptic feedback from squishing a ball and the sound effect of a music box are physical examples of this output. In the virtual world, the sensory output is usually rendered in the form of a visual effect that cannot be perceived in the real world.

3.2 Creating Bidirectional Toy-AR Interactions through Trigger-action Connection.

To simplify the authoring experience, we adopt a trigger-action programming model in MechARspace. In this bidirectional interaction model, a physical toy input can trigger the AR content actions and vice versa. Based on the aforementioned taxonomy of input and output, users can create numerous types of interactions by connecting different triggers to actions. To further increase the diversity of this interaction space, we separate the mapping from input to output as follows:

Discrete trigger-action (Figure 5a) that if the defined trigger event (e.g., a button pressed) occurs, the output actions (e.g., a chest opens) will be activated.

Continuous trigger-action (Figure 5b) that allows users to specify analog relationships between input parameters (e.g., a physical or virtual knob's value) and output parameters (e.g., the opening angle of a chest).

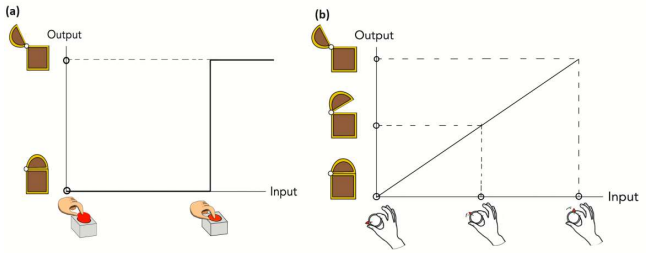

Figure 5: Comparison between discrete (a) and continuous (b) trigger-action that connect input to output.

For discrete trigger-action, we register input as the specific status of the object. For continuous trigger-action, we register input as a series of object statuses. In other words, the trigger-action type is determined by how input is registered. It is worth noting that the collision input cannot be used for continuous trigger-action as it

happens instantly. Moreover, users can connect multiple actions to one trigger to activate them together or connect multiple triggers to one action so that every trigger can activate the same action. MechARspace rejects the connection between the mismatching triggers and actions to ensure valid authoring. For instance, the physics simulation output of the physical toy can only be connected to the collision input of virtual content. To enlarge the interaction space, MechARspace allows users to connect physical input to physical output (e.g., light up the physical LED by pressing a physical button), and virtual input to virtual output (e.g., enlarge a virtual character by turning a virtual knob) that are beyond the scope of virtual-physical joint interactions. Considering the authoring of such unilateral interactions have been thoroughly explored in prior works [61, 62, 66, 71], we will focus on authoring bidirectional interactions in the remaining sections.

In short, our interaction framework consists of three dimensions: ***input entity, output entity,*** and ***trigger-action type***. In the later sections, we will demonstrate the extensiveness of our framework with illustrative use cases. For clear reference back to this framework, we will encode each interaction with the combination of elements from this framework. For instance, Figure 1 shows a user who releases a virtual ghost as he opens the physical chest. This interaction is encoded as **"manipulation (physical) - continuous - global movement (virtual)"**.

4 SYSTEM AND DESIGN

4.1 Modular IoT Toolkit

MechARspace comes with eight IoT modules (Figure 6 a) that can be attached to a traditional toys (Figure 6 b) for them to have AR compatibility. Every module is a standalone device with a battery, a microprocessor, and sensors/actuators built-in, so that users without any circuitry knowledge can use it to prototype AR-enhanced toys.

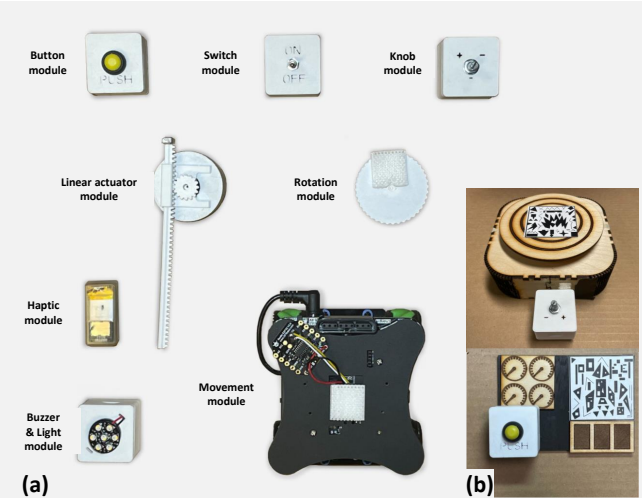

Figure 6: (a) Overview of the IoT modules in MechARspace. (b) Modules attached to toys.

4.1.1 Design Rationale. The design of these modules is guided by the input-output taxonomy of our framework (Section 3). In other words, the role of these modules is to register user input and to facilitate toys' physical output.

We have the "linear actuator module" and the "rotation module" to record the "manipulation" input and to create "local movement" output on the toy,. These two modules have share a unique structure where the servo is connected to the encoder so that we can detect the rotation degree of the servo. This design has two benefits. First, users can use these modules to record their manipulations which are used to animate virtual content accordingly. On the other hand, when users want to program output behaviors, this design makes programming by demonstration (PbD) possible as it requires recording users' demonstrated behaviors [66, 71].

In addition, we have the button module, the switch module, and the knob module that correspond to the "UI element" input. The "movement module" equipped with omni wheels is designed to facilitate the "global movement" output. For users to create "sensory effect" output, we provide the "haptic module" which gives vibration feedback and the "buzzer & light" module which makes sounds and flashes lights. The relationship between the input/output taxonomy and these modules is summarized in Table 1.

Table 1: Relationship between the modules and the input/output taxonomy.

Input	**Module name**
UI element	Button module, switch module, knob module
Manipulation	Linear actuator module, rotation module
Output	**Module name**
Local movement	Linear actuator module, rotation module
Global movement	Movement module
Sensory effect	Haptic module, buzzer & light module

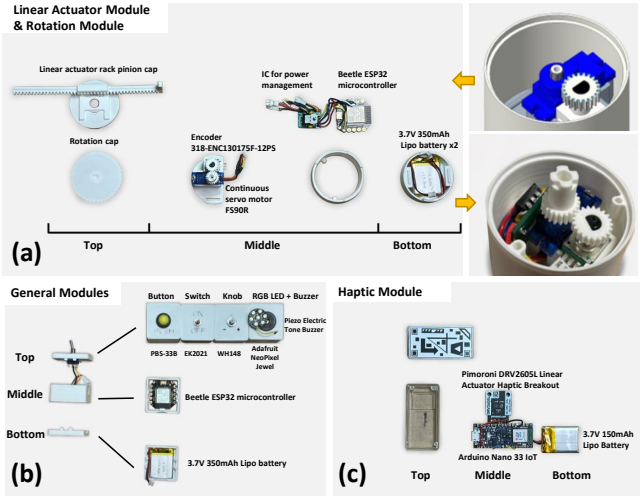

Figure 7: Components and internal structures of the modules.

4.1.2 Components and Internal Structures. Except for the "movement module" which is purchased online [15], other modules are

custom designed and fabricated by us. The case of each module is 3D-printed to encapsulate its electronic components. Each module can be separated into three layers. The top layer is the user interface layer which has physical structures for users to push, rotate, press, and etc. The middle layer contains different electronics including the microcontroller, sensors, and actuators. The bottom layer only holds the battery so that it could be swapped in and out easily. Detailed information on these electronic hardware is depicted in Figure 7.

4.1.3 Connection Mechanism. Each module is a standalone unit with WiFi capability. We connect them to the AR headset (i.e., Hololens 2) through a server running the MQTT protocol (Figure 8), which is the standard for IoT communications [72]. This protocol is based on the publish-subscribe model so that users do not need to reconfigure the network whenever a new module comes in, which enables a more spontaneous design process.

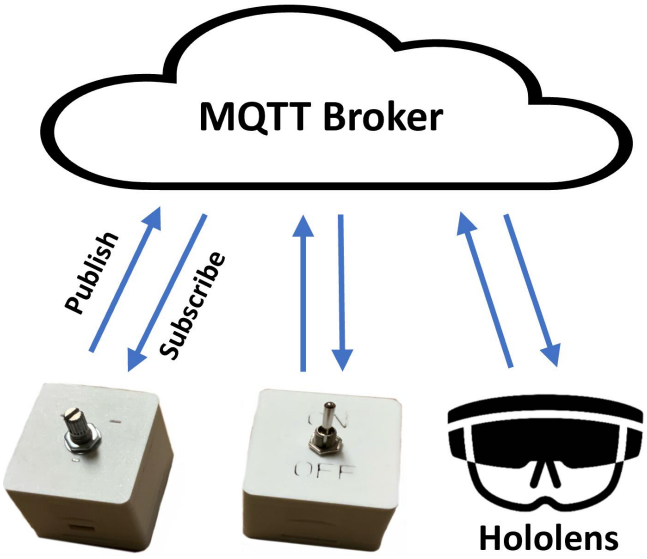

Figure 8: Communication between toolkit and Hololens 2.

4.2 Authoring Interface

In this section, we introduce the interface of MechARspace. The interface is separated into three modes, an **Import Mode** for users to import physical and virtual content, an **Editor Mode** for users to design and edit toy-AR interactions, and a **Play Mode** for users to visualize the authored AR application. To better understand the MechARspace workflow, we provide the following two examples.

Consider a user who wants to create a self-opening physical chest with a virtual lock (Figure 9). First, he/she attaches the physical rotation module to the chest and imports it to the scene along with the virtual model of the lock (Figure 9 a). The user also needs to import the model of the physical chest if he/she intends to anchor the virtual lock to the chest.

After importing the virtual and the physical content, the user can start editing their behaviors. The user selects the input node associated with discrete manipulation on the lock and registers it at the desired locked position (Figure 9 b). Meanwhile, the user selects the output node on the rotation module and demonstrates its behavior by manually opening the box (Figure 9 c). The user can finish the authoring process by connecting recorded input with output through visual programming (Figure 9 d). Finally, the user can enter the **Play Mode** and play with the chest, which will automatically open when the lock is set to the predefined position (Figure 9 e). This interaction can be encoded as **"manipulation (virtual) - discrete - local movement (physical)"**.

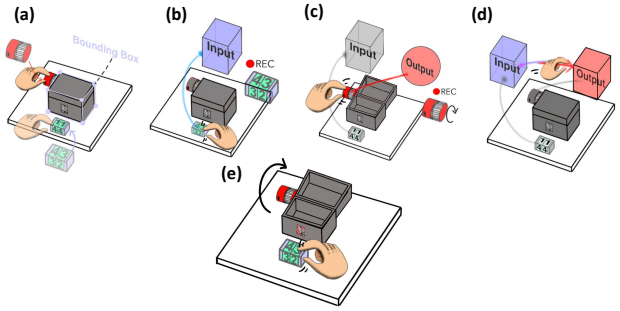

Figure 9: Workflow for an authoring task where the user manipulates a virtual lock and the door will open by itself. (a) Import the virtual and the physical content. (b) Demonstrate the input behavior by setting a lock to a given position. (c) Demonstrate the output behavior by manually opening the chest. (d) Connect the input with the output. (e) Test and play with the authored AR application.

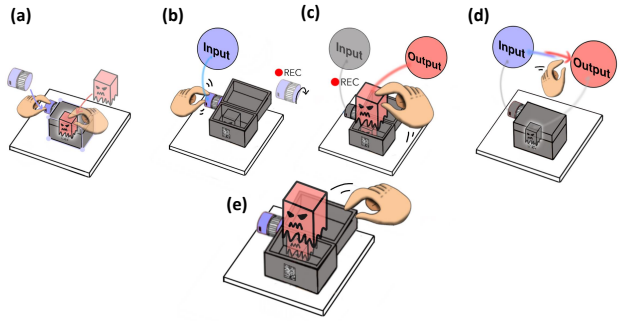

Figure 10: Workflow for an authoring task where the user manually opens a chest and a ghost pops up. (a) Import the virtual and the physical content. (b) Demonstrate the input behavior by recording the start and end value of the attached rotation module while the user opening the chest. (c) Demonstrate the output behavior by moving the ghost upward while scaling it up. (d) Connect the input with the output. (e) Test and play with the authored AR application.

In the second example, a user wants to create an AR version of the traditional toy "Jack-in-the-Box" (Figure 10), where the virtual ghost has to gradually rise and scale up as the user opens the box. Then, it has to scale down and fall back as the box is closed by

the user. The import process is similar to the first case. In the Edit Mode, the user needs to select manipulation as input on the chest and demonstrates the entire process where the box is opened from zero to certain degrees (Figure 10 b). These series of input values continuously map to the virtual ghost's global movement output, which is demonstrated by dragging its scale handle and holding it upwards simultaneously (Figure 10 c). This interaction can be encoded as "**manipulation (physical) - continuous - global movement (virtual)**". In the rest of this section, we will describe the operations of each step in detail.

4.2.1 Import Virtual and Physical Content. To import a physical toy to the scene, the user needs to attach a marker around it and define its bounding box (Figure 11 a). To import virtual content, the user first needs to select it from our library, which includes a variety of pre-created virtual models (Figure 11 b). Then, the user anchors it to a physical object (Figure 11 c). This step is optional as the virtual content can be independent and does not always follow the movement of any physical object. Finally, if an IoT module is mounted on the toy, the user must manually register that information to the system by selecting the hardware module from the toolkit library (Figure 11 d).

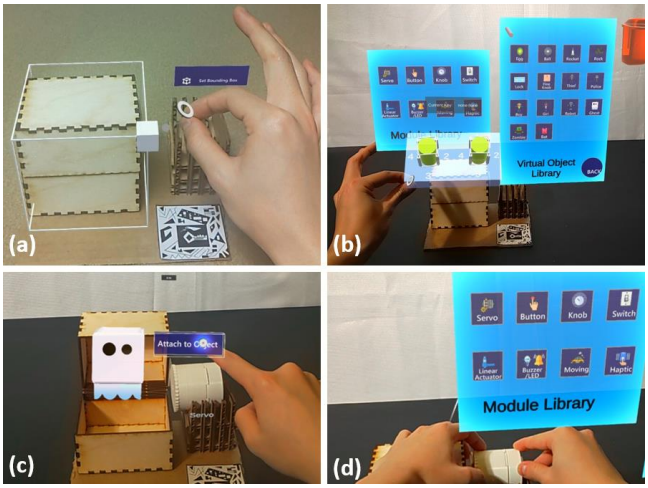

Figure 11: Interface in the import mode. (a) Draw a bounding box around the toy. (b) Select the virtual object from library. (c) Anchor the virtual content to the toy. (d) Import a module from the toolkit library.

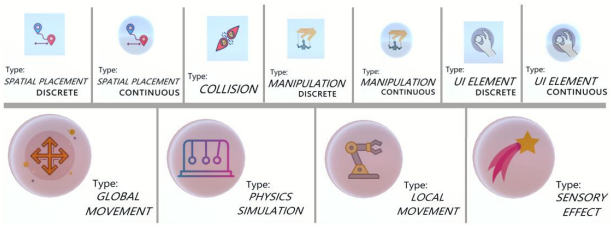

Figure 12: Different types of input (top row) and output (bottom row) supported by MechARspace.

4.2.2 Demonstrate Input Behaviors. As discussed in our framework, MechARspace enables users to demonstrate four types of input: spatial manipulation, collision, manipulation, and UI element (Figure 12 top). Except for collision, the other three types of input can be both discrete and continuous. The user demonstrates the input by first dragging out the corresponding input node and connecting it to the associated object. For collision input and spatial placement input–which involves the interaction between two objects–the user has to draw another connection between them. Except for the collision input–which does not leave room for customization–the user has to demonstrate the input behaviors explicitly. In the interface, the input type is represented visually with a sphere (continuous) or a square (discrete). If the input is discrete, the user only needs to move the object to its desired status and click the button to register (Figure 13 a). In that case, only that specific status is recorded. For continuous input demonstration, the user needs to first move the object (e.g., a knob) to the starting status and register it once (Figure 13 b-1). Then, he/she moves the object to the final status and registers it again (Figure 13 b-2). As a result, a series of values between the start point and the end point are linearly derived (e.g., knob values in the range between two positions).

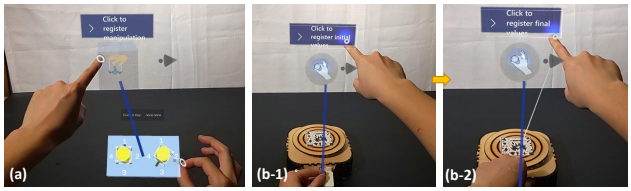

Figure 13: Demonstrate behaviors for (a) discrete input by registering status once and (b) continuous input by registering the start (b-1) and end (b-2) status.

4.2.3 Demonstrate Output Behaviors. As been discussed in our framework, MechARspace enables users to demonstrate four types of output: global movement, local movement sensory effect, and physics simulation (Figure 12 bottom). Users first need to create output nodes for the corresponding objects. In our context, each output is a sequence of events. Both local and global movement can be demonstrated by directly moving, rotating, or scaling the virtual/physical object by hand (Figure 14). Once the movement demonstration is complete, the user can preview its outcome by sliding the progress bar.

We compile the common sensory output to include in our library. Most of them have one or several parameters available for customization purposes. For example, a projectile effect can be customized in size, speed, amount, and gravity. The user needs to set start and end status of this effect by changing its parameters (Figure 15). "1" and "2" on the slider bar represent the parameter value at the start and the end of the output, respectively. By demonstrating its parameter value at start and end, the user can create a sequence of events for output. Some sensory effects such as mesh explosions happen momentarily and do not allow users to set start and end status. As a result, they can only be connected to discrete input. For the physics simulation output, users do not have any room for

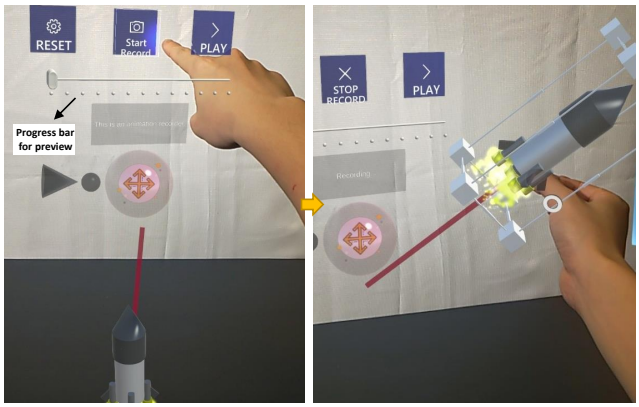

Figure 14: Demonstrate object movement by direct manipulation: the user holds the virtual rocket along a trajectory and record this process.

customization, and hence the output behavior is determined by the collision input physics in real-time.

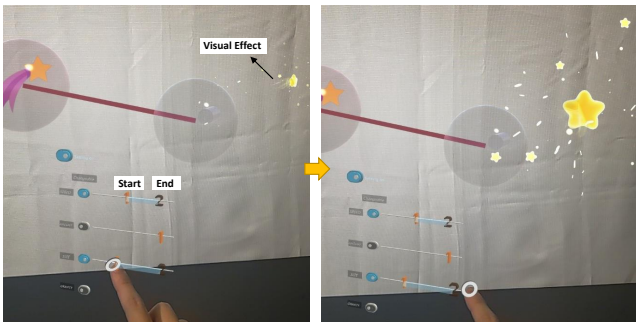

Figure 15: Demonstrate sensory effect output by tuning parameters. "1" and "2" represent the parameter value at the start and the end of the output, respectively.

4.2.4 Program Toy-AR Interactions. The last step of the authoring process is to connect the input with the output, which can be done simply through drawing a line between the input node and the output node. As been discussed in the framework, the trigger-action type is determined by whether the input is discrete or continuous. If the user connects the output with a continuous input, the output's shape stays as a sphere, signaling a continuous trigger-action (Figure 16 a). If the user connects an output with a discrete input, the output turns to a square shape (Figure 16 b), indicating a discrete trigger-action.

4.2.5 Test the AR Applications. **Play Mode** supports users to try out the interactive contents on-the-fly. In this mode, the system keeps tracking all the triggers authored by the user. This process is facilitated through reading sensor data from the IoT modules (e.g., current angle of the rotation module). Moreover, in **Play Mode**, all the trigger and action icons are hidden to give the user an unobstructed view (Figure 1 c).

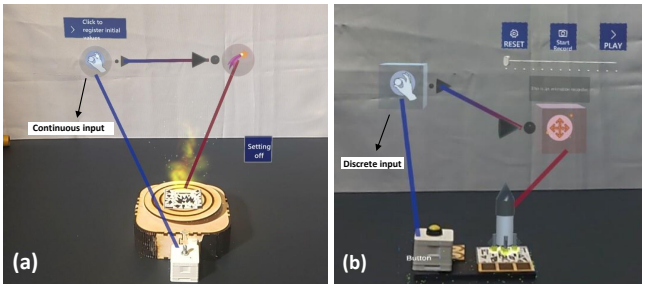

Figure 16: The connection for (a) continuous trigger-action and (b) discrete trigger-action.

4.3 Implementation

We build our system on Hololens 2 [3] using Unity3D (2020.3.0.f1) [11]. The MechARspace user interface is implemented with support from the Microsoft Mixed Reality Toolkit (MRTK) [8]. The tracking of the fiducial markers in the physical toys is done through Vuforia Engine [13]. The physics simulation of virtual content in the real world is enabled using the scene-understanding capability of Hololens 2 and Unity's in-built physics engine. The virtual models and visual effects were downloaded from the Unity asset store [9] and then imported into the system. The real-time sharing of virtual content among multiple users wearing Hololens is supported through through Photon Unity Networking (PUN) [5]. The MQTT broker which handles the data transfer between the IoT toolkit and Hololens 2, runs on a PC (Intel Core i7-8700K, 3.7GHz CPU, 64GB RAM, NVIDIA RTX2080Ti GPU) connected to the local area network.

5 APPLICATION SCENARIOS

5.1 Storytelling Toys

Storytelling is often referred as the process of creating or engaging in narrative structures, which can be a powerful tool for building skills in communication, collaboration, creativity, and retention. Recent works [33, 92] have explored the possibilities of applying AR-enhanced toys as a way for users to physically construct their stories by manipulating toys. At the same time, users could enact their stories by controlling virtual characters. Figure 17 describes a straightforward story: *"Bob is an engineer who designed a security system for his house. If a family member arrives at night, the door automatically opens. One day, a thief tried to break in. The security system worked as planned and triggered the alarm. Initially, the thief still wanted to take his chances, but the frequency of the alarm got higher and higher and the warning light blinked as he approached the door. He eventually had to ran away."* The details of the authoring process and interaction encoding is described in Figure 17. Currently, MechARspace only supports a simplified version of storytelling as its lack the functionality of replaying text and voice, which could be implemented in a future iteration of the system.

5.2 Transforming Daily Objects into Toys

Kudrowitz et al. [55] explains that any tangible item that supports playfulness can be defined as a toy. With our plug-and-play toolkit, our system supports users' spontaneous transformation of daily

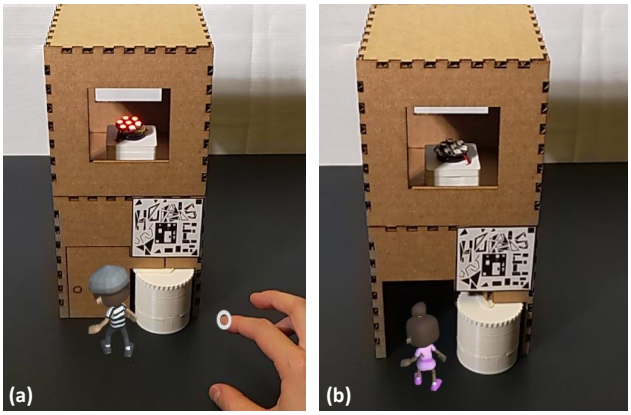

Figure 17: House with a security system. (a) The user utilizes the character's location as the input to trigger the rotation of the door as the output. Interaction encoding: *"Spatial placement (virtual) – discrete - local movement (physical)"*. (b) The user utilizes the character's distance to the house as input to continuously increase the frequency of the buzzer and to blink the LED in red. Interaction encoding: *"Spatial placement (virtual) - continuous – sensory effect (physical)"*.

objects as a way to entertain everyday life with their imagination. For instance, a parent can attach a marker to a fork and use it to tap the hamburger the child is having for lunch. Once the fork collides with the hamburger, it triggers a virtual halo effect (Figure 18 a), which makes the dining experience more enjoyable. With the linear actuator module attached to the lid and bottom of a cookie jar,a person can linearly control virtual ghost size by moving the lid (Figure 18 b).

5.3 Making Miniature Toys More Realistic

In many cases, the toys are simply the miniature replica of their real-world counterpart (e.g., toy trains, toy boats, toy cars). However, some real-world effects cannot be easily reproduced on these toys, thus reducing the realism (e.g., smoke coming out from the train) while playing. On the other hand, AR can fill this gap by simulating these events through visual effects. Figure 19 illustrates a fire truck toy in which the user controls the height of its water gun by manipulating a virtual slider (Figure 19 a). Meanwhile, the user can push a button module to spray virtual water (Figure 19 b). Inside the toy building near the truck, each floor is placed with a buzzer & light module that initially lights up, indicating a fire (Figure 19 b). If the water is sprayed on that floor, the light will dim out, which implies the fire has been extinguished (Figure 19 c).

Robot-shaped toys which provide multiple degrees of freedom for manipulation can also be made more realistic by virtual effect. For example, when the robot's left arm (integrated with the rotation module) reaches out, the user sets it to emit a virtual laser beam (Fig 20 a-1). When the robot arm is raised, the user sets it to initiate a virtual shield (Fig 20 a-2). Meanwhile, the user controls the physical arm of the robot (equipped with linear actuator) by moving a virtual fist in different directions (Fig 20 b).

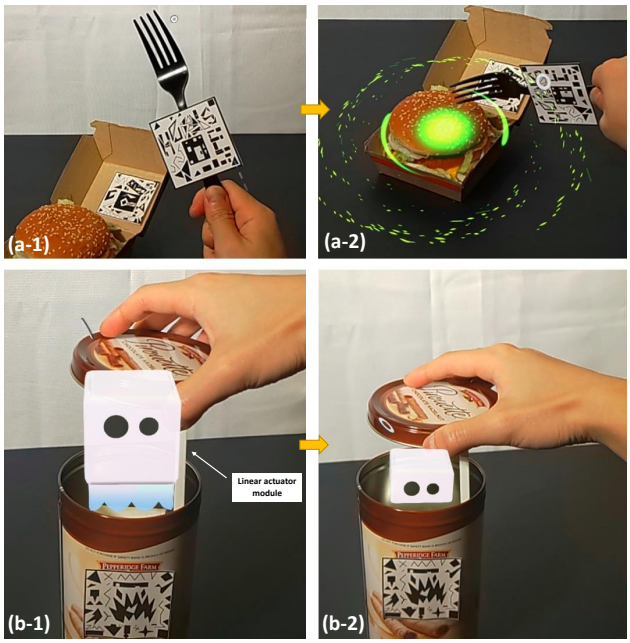

Figure 18: Parents can turn daily objects into playful items for their children. (a-1) The user taps the real hamburger with a fork and (a-2) triggers a virtual halo effect. To author this interaction, he/she uses the collision between the fork and hamburger as the input to trigger the visual effect as the output. Interaction encoding: *"Collision (physical) – discrete - sensory effect (virtual)"*. (b-1) The user attaches the two ends of a linear actuator module to the bottom and the lid of a cookie jar. (b-2) By opening the lid, the user can dynamically change the size of a virtual ghost. To author this interaction, he/she uses the linear actuator module's current length as the input to trigger the global movement of the ghost. Interaction encoding: *"Local movement (physical) - continuous – global movement (virtual)"*.

5.4 Multiplayer Toy Games

A study has shown that multiplayer games could be more fun and engaging than single-player ones [47]. Besides the pure entertainment value, multiplayer games (e.g., board games) offer players opportunities to connect and socialize with others. Meanwhile, the immersive AR experience can be shared with multiple people simultaneously. We propose two multiplayer games with AR-enhanced toys that can be authored with MechARspace.

In the first game, each user controls a virtual soccer player to move around the field. If a virtual player hits a physical soccer ball equipped with movement module, the ball will move in the other direction (Figure 21 a-1). Once the soccer ball passes the goal line, it will trigger a virtual halo effect (Figure 21 a-2).

In the second game, one user has a physical 3D printed canon with a switch module, while another has a physical basket and a haptic module (Figure 21 b-1). The first user can toggle the switch module to fire either virtual stars or virtual bombs. If a virtual

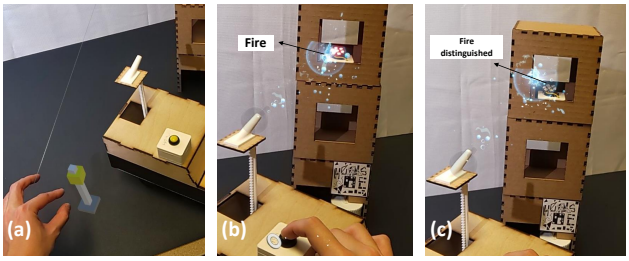

Figure 19: Physical toy fire truck with adjustable-height water gun. (a) The user operates the virtual slider to control the height of the water gun, which is mounted on top of a linear actuator module. To author this interaction, he maps a range of values on the slider to a range of values on the linear actuator module. Interaction encoding: *"UI element (virtual) – continuous - local movement (physical)"*. (b) The user presses the button on the fire truck as the input to spray virtual water. Interaction encoding: *"UI element (physical) - discrete – sensory effect (physical)"*. (c) If the virtual water collides with the physical building, the buzzer & light module–which indicates a fire on that floor—will turn off. For this interaction, he/she uses the collision between building and virtual water to trigger the "turn off" action of the buzzer & light module. Interaction encoding: *"Collision (virtual) - discrete – sensory effect (physical)"*.

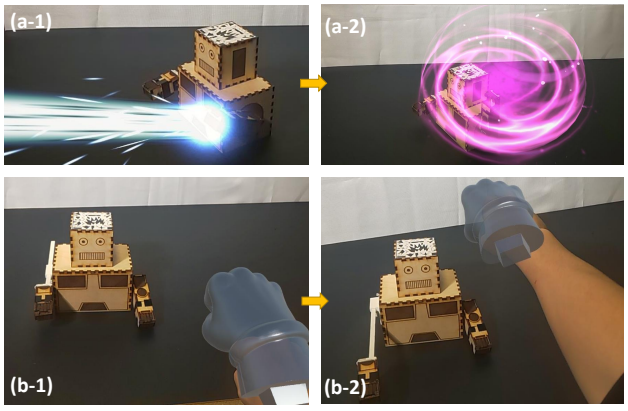

Figure 20: Robot that can both be controlled and move on its own: (a) The user adjusts the robot's parts to different poses and associates them with different visual effects. Interaction encoding: *"Local movement (physical)– discrete - sensory effect (virtual)"*. (b) The user sets the robot to perform the 'punch' action when the virtual fist moves forward. Interaction encoding: *"Spatial movement (virtual) - discrete - local movement (physical)"*.

bomb hits the basket, it triggers the activation of the haptic module (Figure 21 b-2) on the basket held by the second user.

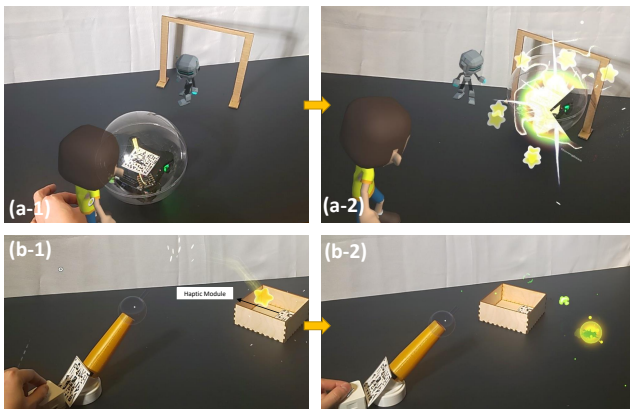

Figure 21: Multiplayer games (a-1): Two users manipulate their own virtual characters to play a soccer game. If either character collides with the physical ball, the ball will move in the opposite direction. Interaction encoding: *"Collision (physical)– discrete - physics simulation (virtual)"*. (a-2) If the physical ball crosses the goal line, a virtual halo effect will be triggered. Interaction encoding: *"Spatial movement (physical) - discrete -sensory effect (virtual)."* (b-1) As the user toggles the switch, the physical cannon shoots two kinds of virtual effects. Interaction encoding: *"UI element (physical) - discrete - sensory effect (virtual)"*. (b-2) If specific visual effect collides with the chest, the haptic module attached will start vibrating. Interaction encoding: *"Collision (virtual) - discrete -sensory effect (physical)"*.

6 USER STUDY EVALUATION

We evaluate MechARspace's overall system usability and its efficacy as a design tool. Six users were recruited (five males and one female, aged 19-26). They have all experienced AR applications on either cell phones, tablets, or head-mounted devices. Four of our users have participated in toy making activities in some forms. One of them had taken the CAD-related classes before. Another one of them once have once followed an online tutorial to fabricate a customized chessboard for himself. None of them is a professional AR/VR designer or programmer, considering that MechARspace is designed as an end-user authoring tool. None of the users had experienced our system before conducting the user study. The entire study took around 3 hours, and each user was paid 20 dollars. The study took place at our lab equipped with common digital fabrication tools such as a 3D printer and a laser cutter. We first asked the users to walk through the Hololens 2 official tutorial to learn how to navigate the user interface with basic hand gestures. After the first session, the user completed a survey with Likert-type (scaled 1-5) questions regarding the user experience of specific system features and a standard System Usability Scale (SUS) questionnaire. After the second session, we conducted an open-ended interview to get subjective feedback on our system.

6.1 Session One: System Usability Evaluation

We designed six micro tasks for the first user study session (Figure 22). Each task contains one pair of toy-AR interaction for users

Table 2: Detailed descriptions on the tasks in the first session.

Task	Description	Module	Interaction Encoding
T1	A user wields a physical hammer to smash a virtual golden egg	None	Collision - discrete - sensory
T2	A user presses a physical button on the control panel to launch a virtual rocket	Button module	UI element - discrete - global movement
T3	A user turns a physical knob to adjust the intensity of the virtual flame on the stove	Knob module	UI element - continuous - sensory
T4	As a virtual thieve reaches the house, the frequency of physical buzzer increases	Buzzer & light module	Spatial placement - continuous - sensory
T5	A user throws a virtual rock at the physical car to send it backwards	Movement module	Collision - discrete - physics simulation
T6	A user changes the combination of a virtual lock to open the door	Linear actuator module	Manipulation - discrete -local movement

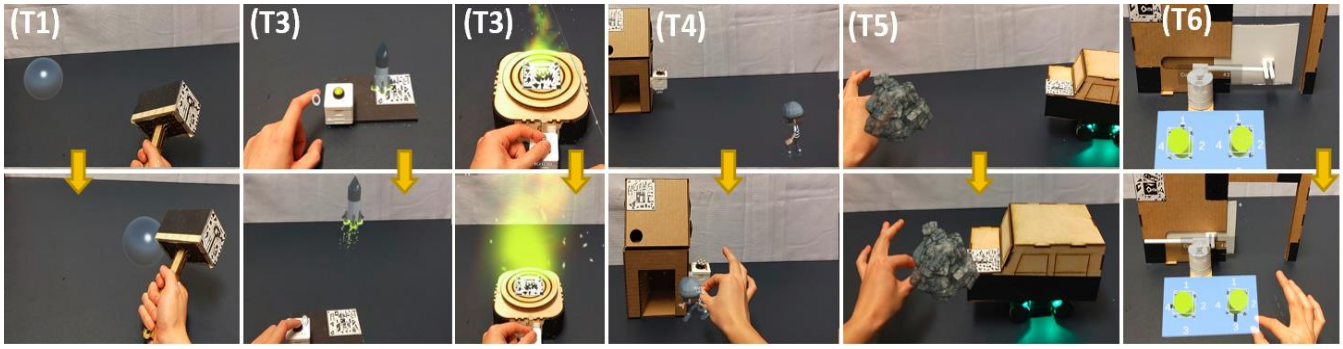

Figure 22: Tasks in the first session.

to author. Half of these interactions (Task 1-3) use the toy as the trigger to actuate the AR content, and the other three (Task 4-6) use the AR content to actuate the toy. Our input and output categories are all covered in these interactions. Both continuous and discrete trigger-action types are included as well. The description of each task is detailed in Table 2. The goal of this session was to evaluate the usability of the MechARspace system and to explore the user experience of authoring toy-based AR applications inside a mixed reality environment.

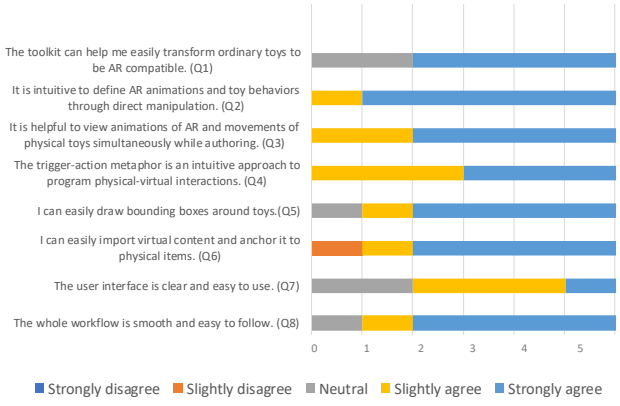

Figure 23: Likert-type questionnaire results of the first session.

6.1.1 Result and Discussion. The Likert-type question ratings are shown in Figure 23. In general, users found the workflow easy to follow (Q8: avg=4.5, sd=0.83), and the interface clear (Q7: avg=3.83, sd=0.75). *"Once I walked through the tasks on the first round, I was confident I could perform some customization all by myself (P3)"*. The prerequisite procedures for loading virtual (Q6: avg=4.33, sd=1.21) and physical (Q5: avg=4.5, sd=0.83) objects into the scene were also considered straightforward. Most of them acknowledged that the trigger-action metaphor was suitable for toy-based AR application creation (Q4: avg=4.5, sd=0.54).*"The action-reaction type of programming logic was super easy to follow, and I feel it will cover most scenarios (P1)"*. Meanwhile, users appreciated the immersive authoring environment, which allows them to concurrently view the toy and its associated AR animations (Q3: avg=5, sd=0). *"I like the fact that I can view the toy and AR at the same time, giving me a look at the whole picture (P6)"*. Also, they found that programming by demonstration could greatly streamline the programming process (Q2: avg=4.83, sd=0.41). *"There is nothing more intuitive to define the behaviors [of virtual and physical objects] than directly manipulating them (P5)"*. All users were in favor of the plug-and-play IoT toolkit we provided to enrich the AR experience (Q1: avg=4.83, sd=1.03). *"I felt like I could simply attach it to many ordinary things and animate them in AR (P6)"*. During the user study, we observed that most users (five out of six) struggled when they initially started learning how to use hand gestures to navigate the AR interface in Hololens. Two of them attributed this difficulty to the lack of haptic feedback when interacting with Hololens's virtual menus. Similar observations have been made in prior works which also adopt Hololens as the testing platform [81, 93]. However, learning to use the system itself went smoothly once they passed that threshold. the standard SUS survey result for the entire study received 83 out of 100 with a standard deviation of 10.58, which indicated high usability of the whole system.

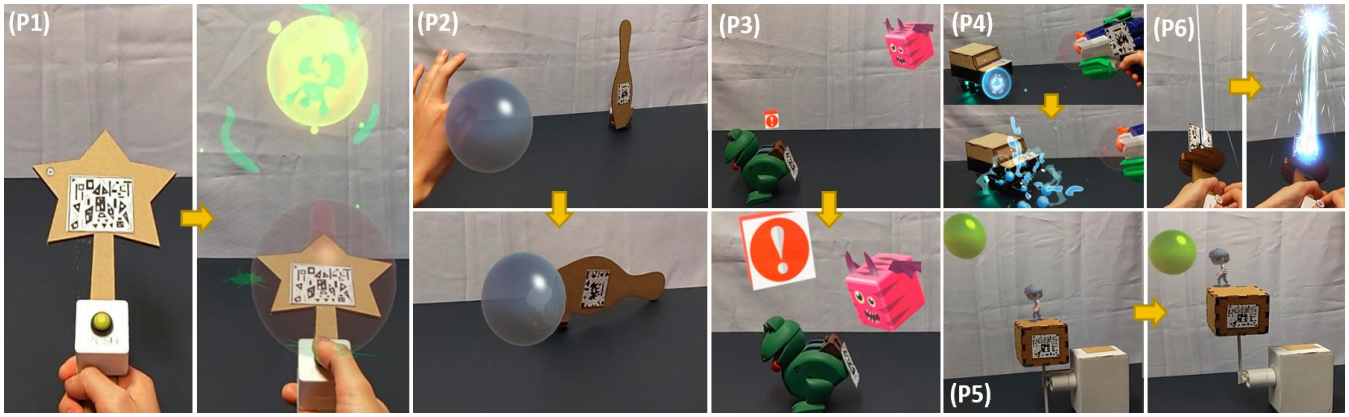

Figure 24: AR-enhanced toy design outcomes from the second session of user study.

Table 3: Detailed descriptions on the AR-enhanced toys made by users in the second session.

Participants	Description	Module	Interaction Encoding
P1	A user presses the physical button on the magic wand to cast a spell.	Button module	UI element - discrete - sensory effect
P2	A user throws a virtual ball to strike down the physical bowling pin.	Rotation module	Collision - discrete - local movement
P3	As a user grabs a toy frog closer to the virtual demon, the danger sign will become larger.	None	Spatial placement - continuous - global movement
P4	A user adapts a traditional water gun to let it spray virtual water. He/she can then spray water on a physical car to push it back.	Button module Movement module	UI element - discrete - sensory effect Collision - discrete - physics simulation
P5	If a virtual character controlled by a user jumps on a platform, he will be elevated to reach the golden egg.	Rotation module	Spatial placement - discrete - local movement
P6	A user holds the hilt of a saber, turns the knob to change the intensity of virtual saber effect.	Knob module	UI element - continuous - sensory effect

6.2 Session Two: Design and Implement AR-enhanced Toys

We sought to examine MechARspace in a freestyle DIY session. In this session, we began by asking participants to capture, design, and implement their ideas for an AR-enhanced toy. This brainstorming and quick-prototyping of the toy took about 80 minutes. At this stage, participants were already familiar with the system, but we walked them through all the available potential toy-AR joint interactions by explaining to them the interaction model we described in section 3. A researcher was present to clarify any questions the participant may have. Participants were given the freedom paper-prototype their designs, implement them using fabrication tools (e.g., laser cutter) inside our laboratory, and then enhance the toys by using MechARspace to customize the toy-AR interactions. If their intended virtual models or visual effects were not in our library, we would download and import them to the library. With the fabricated toys and these virtual content, users could finally validate their designed joint interaction with MechARspace. In the end, we held an in-depth discussion with our users regarding their perception of empowering a DIY-ed toy with AR technology in general.

6.2.1 Result and Discussion. Figure 24 and Table 3 showcase the toys created by the users during this open creation session. Each user's toy consists of at least one pair of physical-virtual interaction.

The purpose of our study is to validate a unified framework that can be useful to novice designers and DIY-ers. Overall, participants found the interaction model to be comprehensive enough that they were quickly able to capture their ideas and prototype them. *"Initially, I thought designing the virtual aspect of my toy–and the mechanical part too–would be much more difficult and need more tries, but these UIs really give me a good starting point(P4)".*

Another common theme across participants was the understanding of the motivation and the significance behind the system. *"I can definitely see the future of toys, where they are unlimited by physical constraints... This will bring about a new relationship with the environment and the virtual parts of it (P6)".*

One of the participants was enthusiastic about the prospect of using the system for storytelling as she works with children. *"The AR increases the dimensions and possibilities of the toy. I can think of a scenario where I give each kid an action figure of their choice. Then, with the AR and the modules, I get them swords, hammers, shields.Then, they go to fight giant spiders and monsters. Now, this is a full-blown adventure (P3)".*

An important concern that was addressed with regard to the participants' non-technical backgrounds is that programming AR applications no longer seems unattainable. *"I used to think programming AR would be complicated. But this input-output model you proposed really simplifies a lot of things make the whole process less daunting. I feel like I can just create a program that just connects lines (P1)".* Participants gave emphasis on the usefulness of using

AR content enabled by fast customization: *"I can download virtual content, instantly try it out without further delay. If I am not satisfied with it I can simply delete it and try another. (P2)"*. Conversely, the virtual content hovering on or close to the physical toy made it simpler and more intuitive for participants to use as guiding points and thus, allowing them to *"have the benefits of both worlds (P5)"*.

7 LIMITATIONS AND FUTURE WORKS

7.1 The Ceiling of the Interaction Model

The proposed input-output interaction model does have its ceilings. First of all, this model assumes the input condition and output event are both predictable so that users can explicitly demonstrate them respectively and link them together. However, there may exist some random events during toy-AR interactions. In the second session of our user study, one user initially wanted to design an AR version of the "whack a mole [14]" toy where he could hold a physical hammer to smash virtual moles that pop up randomly from the ground. In that case, the random behaviors of the mole cannot be explicitly authored by the user. Mainstream game engines have resorted to machine learning technology (e.g., Unity machine learning agent [10]) to create objects with random behaviors (e.g., NPCs) without predefinitions. An alternative approach is to introduce the random timer as a input that can trigger events. In the future, we will explore the possibility of employing similar technologies in our system.

In addition, MechARspace only supports simple and direct mappings between triggers and actions. Some complex mappings such as condition (an action only reacts to a trigger given some preconditions, such as location, time, or other triggers) or parallel (an action only reacts when multiple triggers are activated concurrently). While most toys are straightforward to play with, some others (e.g., puzzle cards) do have intricate game logic. One possible solution would be to introduce a more comprehensive visual programming logic similar to the ones in CAPturAR [82], Ivy [31], and FlowMatic [91]. For example, the "loop" construct introduced in Ivy [31] could be applied to program a virtual character who performs some routine tasks.

7.2 Creation and Integration of Virtual & Physical Content into the Scene

Currently, the virtual models and visual effects of MechARspace are preloaded into the library. Future iterations of MechARspace should support real-time import of virtual content from popular online platforms (e.g., Unity asset store [9]) to support more spontaneous creations. For a physical content, we asked users to integrate it into the scene by attaching a marker and specifying its bounding box. Such a process could be streamlined by scanning the geometry of physical objects in real-time. However, the scanning precision of Hololens 2 is too low for this task. In the future, we will explore the possibility of using external devices (e.g., depth camera [4]) to scan the model and sending it directly into the Hololens 2.

7.3 Crowding of Virtual Content

Right now, the UI elements are displayed in-situ besides the virtual content. While most of the users appreciated this feature since *"the spatial relationship between them brings more logical sense to the creation process (P2)"*. Some of them also expressed concerns that too many overlapping virtual content would inevitably crowd the interface if several interactions need to be authored concurrently. One possible solution to this issue would be to display the UI elements adaptively. For instance, we can decrease the opacity of the trigger-action link that is not currently been authored by the user.

7.4 Tracking Toys

We used fiducial markers attached to the toys to track their real-time positions. This approach is susceptible to occlusion, and the computation capability of Hololens 2 constrains the tracking speed and frequency. During the user study, some of our users expressed frustration when the objects lost track. *"Whenever I move the hammer too quick, it will lose track and interrupt my play. It is really annoying (P2)"*. In the future, we will explore the possibility of designing a dedicated IoT toolkit (e.g., RFID tag [80]) for tracking or employing markerless vision-based tracking technology [29].

7.5 IoT Toolkit

As for now, every module is a standalone device with a battery, a microprocessor, sensors, and actuators integrated, which inevitably makes them bulky, especially when they are attached to small-sized toys. In the future, we will develop a CAD plugin for our toolkit, which would allow users to pre-allocate space for these modules. In this way, the toolkit can be integrated inside instead of outside the toy.

8 CONCLUSION

In this work, we present MechARspace, an authoring tool that enables novice designers and DIY-ers to create AR applications based on their physical toys. MechARspace allows users to program customized toy-AR interactions in-situ by demonstration while using relevant contextual elements as references. We start by compiling the bidirectional interaction model that maps various types of toys' actions to responsive behaviors of virtual contents. Following this interaction model, we design our immersive visual programming interface so that users can author corresponding interaction modalities through simple trigger-action programming. Furthermore, we develop a collection of IoT modules to help designers effortlessly integrate their toys into the AR scene without lengthy electronic prototyping processes. To explore the capability of MechARspace, we demonstrate four groups of applications scenarios. Through a two-session user study, we first proved our system's usability and then its utility as a design tool to help impromptu AR-enhanced toy design. Thus, MechARspace provides the HCI community with a unified framework and an open landscape into future designs of AR authoring tools for toys.

ACKNOWLEDGMENTS

We wish to give a special thanks to the reviewers for their invaluable feedback. This work is partially supported by the NSF under the Future of Work at the Human Technology Frontier (FW-HTF) 1839971. We also acknowledge the Feddersen Distinguished Professorship Funds. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily

reflect the views of the funding agency. We would also like to thank Enze Jiang for his help in writing the alt-text.

REFERENCES

[1] 2022. ARCore Build the Future. https://developers.google.com/ar.

[2] 2022. Dive into the world of augmented reality. https://developer.apple.com/augmented-reality/.

[3] 2022. Hololens2. https://www.microsoft.com/en-us/hololens.

[4] 2022. Intel Depth Camera. https://www.intelrealsense.com/depth-camera-d435/.

[5] 2022. Introduction to the Multi-user capabilities tutorials. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/tutorials/mr-learning-sharing-01.

[6] 2022. LEGO AR studio. https://www.lego.com/en-us/aboutus/news/2019/october/lego-ar-studio/.

[7] 2022. Mario Kart Live. https://vrscout.com/news/mario-kart-live-ar-now-available/.

[8] 2022. MRTK. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05.

[9] 2022. Unity Asset Store. https://assetstore.unity.com/.

[10] 2022. Unity Machine Learning Agents. https://unity.com/products/machine-learning-agents/.

[11] 2022. Unity Real-Time Development Platform. https://unity.com/.

[12] 2022. Unreal Engine: The most powerful real-time 3D creation tool. https://www.unrealengine.com/en-US.

[13] 2022. Vuforia Engine. https://developer.vuforia.com/.

[14] 2022. Whac-A-Mole. https://en.wikipedia.org/wiki/Whac-A-Mole.

[15] 2022. yahboom. https://category.yahboom.net/.

[16] Raphael Anderegg, Loïc Ciccone, and Robert W Sumner. 2018. PuppetPhone: puppeteering virtual characters using a smartphone. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. 1–6.

[17] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-action-circuits: Leveraging generative design to enable novices to design and build circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 331–342.

[18] Takafumi Aoki, Takashi Matsushita, Yuichiro Iio, Hironori Mitake, Takashi Toyama, Shoichi Hasegawa, Rikiya Ayukawa, Hiroshi Ichikawa, Makoto Sato, Takatsugu Kuriyama, et al. 2005. Kobito: virtual brownies. In *ACM SIGGRAPH 2005 emerging technologies*. 11–es.

[19] Rahul Arora, Rubaiat Habib Kazi, Danny M Kaufman, Wilmot Li, and Karan Singh. 2019. Magicalhands: Mid-air hand gestures for animating in vr. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 463–477.

[20] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. 2008. *Survey: Robot programming by demonstration.* Technical Report. Springrer.

[21] Mark Billinghurst, Raphael Grasset, and Julian Looser. 2005. Designing augmented reality interfaces. *ACM Siggraph Computer Graphics* 39, 1 (2005), 17–22.

[22] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. 2019. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–28.

[23] Daniel Calife, João Luiz Bernardes Jr, and Romero Tori. 2009. Robot Arena: An augmented reality platform for game development. *Computers in Entertainment (CIE)* 7, 1 (2009), 1–26.

[24] Sylvain Calinon. 2009. *Robot programming by demonstration.* EPFL Press.

[25] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 521–534.

[26] Yuanzhi Cao, Zhuangying Xu, Terrell Glenn, Ke Huo, and Karthik Ramani. 2018. Ani-Bot: A Modular Robotics System Supporting Creation, Tweaking, and Usage with Mixed-Reality Interactions. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction*. 419–428.

[27] Jon Carroll and Fabrizio Polo. 2013. Augmented reality gaming with sphero. In *ACM Siggraph 2013 Mobile*. 1–1.

[28] Ya-Wen Cheng, Yuping Wang, Yu-Fen Yang, Zih-Kwan Yang, and Nian-Shing Chen. 2020. Designing an authoring system of robots and IoT-based toys for EFL teaching and learning. *Computer Assisted Language Learning* 34, 1-2 (2020), 6–34.

[29] Andrew I Comport, Eric Marchand, Muriel Pressigout, and Francois Chaumette. 2006. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on visualization and computer graphics* 12, 4 (2006), 615–628.

[30] Nathan Delson and Harry West. 1996. Robot programming by human demonstration: Adaptation and inconsistency in constrained motion. In *Proceedings of IEEE International conference on Robotics and Automation*, Vol. 1. IEEE, 30–36.

[31] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, and George Fitzmaurice. 2017. Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In *Proceedings of the 43rd Graphics Interface Conference*. 156–162.

[32] Markus Funk, Oliver Korn, and Albrecht Schmidt. 2014. An augmented workplace for enabling user-defined tangibles. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. 1285–1290.

[33] Terrell Glenn, Ananya Ipsita, Caleb Carithers, Kylie Peppler, and Karthik Ramani. 2020. StoryMakAR: Bringing stories to life with an augmented reality & physical prototyping toolkit for youth. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[34] Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. 209–218.

[35] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S Reynolds, and Joshua R Smith. 2017. Finding common ground: A survey of capacitive sensing in human-computer interaction. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 3293–3315.

[36] Sebastian Günther, Florian Müller, Martin Schmitz, Jan Riemann, Niloofar Dezfuli, Markus Funk, Dominik Schön, and Max Mühlhäuser. 2018. CheckMate: Exploring a tangible augmented reality interface for remote interaction. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–6.

[37] Sunao Hashimoto, Akihiko Ishida, Masahiko Inami, and Takeo Igarashi. 2011. Touchme: An augmented reality based remote robot manipulation. In *The 21st International Conference on Artificial Reality and Telexistence, Proceedings of ICAT2011*, Vol. 2.

[38] Anuruddha Hettiarachchi and Daniel Wigdor. 2016. Annexing reality: Enabling opportunistic use of everyday objects as tangible proxies in augmented reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1957–1967.

[39] Juan David Hincapié-Ramos, Kasim Ozacar, Pourang P Irani, and Yoshifumi Kitamura. 2015. GyroWand: IMU-based raycasting for augmented reality head-mounted displays. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*. 89–98.

[40] Steve Hinske, Matthias Lampe, Nicola Yuill, Sara Price, and Marc Langheinrich. 2009. Kingdom of the knights: Evaluation of a seamlessly augmented toy environment for playful learning. In *Proceedings of the 8th International Conference on Interaction Design and Children*. 202–205.

[41] Steve Hinske, Marc Langheinrich, and Matthias Lampe. 2008. Towards guidelines for designing augmented toy environments. In *Proceedings of the 7th ACM conference on Designing interactive systems*. 78–87.

[42] Takefumi Hiraki, Shogo Fukushima, and Takeshi Naemura. 2016. Phygital field: an integrated field with a swarm of physical robots and digital images. In *SIGGRAPH ASIA 2016 Emerging Technologies*. 1–2.

[43] Donell Holloway and Lelia Green. 2016. The Internet of toys. *Communication Research and Practice* 2, 4 (2016), 506–519.

[44] Steven Houben, Connie Golsteijn, Sarah Gallacher, Rose Johnson, Saskia Bakker, Nicolai Marquardt, Licia Capra, and Yvonne Rogers. 2016. Physikit: Data engagement through physical ambient visualizations in the home. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1608–1619.

[45] Gaoping Huang, Xun Qian, Tianyi Wang, Fagun Patel, Maitreya Sreeram, Yuanzhi Cao, Karthik Ramani, and Alexander J Quinn. 2021. Adaptutar: An adaptive tutoring system for machine tasks in augmented reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[46] Ke Huo and Karthik Ramani. 2017. Window-shaping: 3d design ideation by creating on, borrowing from, and looking at the physical world. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*. 37–45.

[47] Ioanna Iacovides, Anna L Cox, Ara Avakian, and Thomas Knoll. 2014. Player strategies: Achieving breakthroughs and progressing in single-player and cooperative games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*. 131–140.

[48] Hiroshi Ishii. 2008. The tangible user interface and its evolution. *Commun. ACM* 51, 6 (2008), 32–36.

[49] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. 234–241.

[50] Kevin Sebastian Kain, Susanne Stadler, Manuel Giuliani, Nicole Mirnig, Gerald Stollnberger, and Manfred Tscheligi. 2017. Tablet-based augmented reality in the factory: Influence of knowledge in computer programming on robot teaching tasks. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. 151–152.

[51] Shunichi Kasahara, Ryuma Niiyama, Valentin Heun, and Hiroshi Ishii. 2013. exTouch: spatially-aware embodied manipulation of actuated objects mediated by augmented reality. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*. 223–228.

[52] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 395–405.

[53] Annie Kelly, R Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. 2018. ARcadia: A rapid prototyping platform for real-time tangible interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–8.

[54] Joo Chan Kim, Teemu H Laine, and Christer Åhlund. 2021. Multimodal interaction systems based on internet of things and augmented reality: A systematic literature review. *Applied Sciences* 11, 4 (2021), 1738.

[55] Barry M Kudrowitz and David R Wallace. 2010. The play pyramid: A play classification and ideation tool for toy design. *International Journal of Arts and Technology* 3, 1 (2010), 36–56.

[56] David Kurlander, Allen Cypher, and Daniel Conrad Halbert. 1993. *Watch what I do: programming by demonstration*. MIT press.

[57] Fabrizio Lamberti, Davide Calandra, Federica Bazzano, Filippo G Prattico, and Davide M Destefanis. 2018. RobotQuest: A robotic game based on projected mixed reality and proximity interaction. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE, 1–9.

[58] Gun A Lee, Gerard J Kim, and Mark Billinghurst. 2005. Immersive authoring: What you experience is what you get (wyxiwyg). *Commun. ACM* 48, 7 (2005), 76–81.

[59] Gun A Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. 2004. Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 172–181.

[60] Jakob Leitner, Michael Haller, Kyungdahm Yun, Woontack Woo, Maki Sugimoto, Masahiko Inami, Adrian David Cheok, and HD Been-Lirn. 2010. Physical interfaces for tabletop games. *Computers in Entertainment (CIE)* 7, 4 (2010), 1–21.

[61] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2021. Rapido: Prototyping Interactive AR Experiences through Programming by Demonstration. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 626–637.

[62] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid augmented reality video prototyping using sketches and enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[63] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-user development: An emerging paradigm. In *End user development*. Springer, 1–8.

[64] Richard G McDaniel and Brad A Myers. 1999. Getting more out of programming-by-demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 442–449.

[65] Leon Müller, Ken Pfeuffer, Jan Gugenheimer, Bastian Pfleging, Sarah Prange, and Florian Alt. 2021. Spatialproto: Exploring real-world motion captures for rapid prototyping of interactive mixed reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

[66] Ryosuke Nakayama, Ryo Suzuki, Satoshi Nakamaru, Ryuma Niiyama, Yoshihiro Kawahara, and Yasuaki Kakehi. 2019. Morphio: Entirely soft sensing and actuation modules for programming shape changes through tangible interaction. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. 975–986.

[67] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. XRDirector: A role-based collaborative immersive authoring system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[68] Michael Nebeling and Maximilian Speicher. 2018. The trouble with augmented reality/virtual reality authoring tools. In *2018 IEEE international symposium on mixed and augmented reality adjunct (ISMAR-Adjunct)*. IEEE, 333–337.

[69] Gary Ng, Joon Gi Shin, Alexander Plopski, Christian Sandor, and Daniel Saakes. 2018. Situated game level editing in augmented reality. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction*. 409–418.

[70] Xun Qian, Fengming He, Xiyun Hu, Tianyi Wang, Ananya Ipsita, and Karthik Ramani. 2022. ScalAR: Authoring Semantically Adaptive Augmented Reality Experiences in Virtual Reality. In *CHI Conference on Human Factors in Computing Systems*. 1–18.

[71] Hayes Solos Raffle, Amanda J Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 647–654.

[72] Dipa Soni and Ashwin Makwana. 2017. A survey on mqtt: a protocol of internet of things (iot). In *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*, Vol. 20. 173–177.

[73] Andrew Spielberg, Alanson Sample, Scott E Hudson, Jennifer Mankoff, and James McCann. 2016. RapID: A framework for fabricating low-latency interactive objects with RFID tags. In *Proceedings of the 2016 chi conference on human factors in computing systems*. 5897–5908.

[74] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. 2022. Augmented Reality and Robotics: A Survey and Taxonomy for AR-enhanced Human-Robot Interaction and Robotic Interfaces. In *CHI Conference on Human Factors in Computing Systems*. 1–33.

[75] Ryo Suzuki, Jun Kato, Mark D Gross, and Tom Yeh. 2018. Reactile: Programming swarm user interfaces through direct physical manipulation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[76] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. Realitysketch: Embedding responsive graphics and visualizations in AR through dynamic sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 166–181.

[77] Jeff KT Tang and Jordan Tewell. 2015. Emerging human-toy interaction techniques with augmented and mixed reality. In *Mobile Services for Toy Computing*. Springer, 77–105.

[78] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Kylie Peppler, Thomas Redick, and Karthik Ramani. 2020. Meta-AR-app: an authoring platform for collaborative augmented reality in STEM classrooms. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.

[79] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Feiyang Wang, Subramanian Chidambaram, and Karthik Ramani. 2022. ColabAR: A Toolkit for Remote Collaboration in Tangible Augmented Reality Laboratories. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW1 (2022), 1–22.

[80] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, and Haiyan Zhang. 2018. Project zanzibar: A portable and flexible tangible interaction platform. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[81] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 552–567.

[82] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPturAR: An augmented reality tool for authoring human-involved context-aware applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 328–341.

[83] Matt Whitlock, Jake Mitchell, Nick Pfeufer, Brad Arnot, Ryan Craig, Bryce Wilson, Brian Chung, and Danielle Albers Szafir. 2020. MRCAT: In situ prototyping of interactive AR environments. In *International Conference on Human-Computer Interaction*. Springer, 235–255.

[84] Karl DD Willis, Ivan Poupyrev, and Takaaki Shiratori. 2011. Motionbeam: a metaphor for character interaction with handheld projectors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1031–1040.

[85] Ryan Wistort and Cynthia Breazeal. 2011. TofuDraw: A mixed-reality choreography tool for authoring robot character performance. In *Proceedings of the 10th International Conference on Interaction Design and Children*. 213–216.

[86] Hui Ye and Hongbo Fu. 2022. ProGesAR: Mobile AR Prototyping for Proxemic and Gestural Interactions with Real-world IoT Enhanced Spaces. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[87] Hui Ye, Kin Chung Kwan, Wanchao Su, and Hongbo Fu. 2020. ARAnimator: in-situ character animation in mobile AR with user-defined motion gestures. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 83–1.

[88] Neng-Hao Yu, Li-Wei Chan, Seng Yong Lau, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Fang-I Hsiao, Lung-Pan Cheng, Mike Chen, Polly Huang, et al. 2011. TUIC: enabling tangible interaction on capacitive multi-touch displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2995–3004.

[89] Ya-Ting Yue, Yong-Liang Yang, Gang Ren, and Wenping Wang. 2017. Scenectrl: Mixed reality enhancement via efficient scene editing. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 427–436.

[90] Jürgen Zauner, Michael Haller, Alexander Brandl, and Werner Hartman. 2003. Authoring of a mixed reality assembly instructor for hierarchical structures. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* IEEE, 237–246.

[91] Lei Zhang and Steve Oney. 2020. Flowmatic: An immersive authoring tool for creating interactive scenes in virtual reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 342–353.

[92] Zhiying Zhou, Adrian David Cheok, JiunHorng Pan, and Yu Li. 2004. Magic Story Cube: an interactive tangible interface for storytelling. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 364–365.

[93] Fengyuan Zhu and Tovi Grossman. 2020. Bishare: Exploring bidirectional interactions between smartphones and head-mounted augmented reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.