FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

ISAR: An Authoring System for Interactive Tabletops

Zardosht Hodaie, M.Sc.

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Forschungs- und Lehreinheit I
Angewandte Softwaretechnik

ISAR: An Authoring System for Interactive Tabletops

Zardosht Hodaie

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Michael Georg Bader
Prüfer der Dissertation:	1. Prof. Dr. Bernd Brügge
	2. Prof. Gudrun Johanna Klinker, Ph.D.

Die Dissertation wurde am 15.04.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03.08.2020 angenommen.

To my beloved parents.
To Sara. Sat Nam.

Acknowledgements

This work would not have been possible without the support of many people.

First of all I would like to thank my advisor Prof. Bernd Brügge, for his continuous support and providing me the opportunity to grow and gain experience in his group. To me he is not only an advisor, but a teacher and great person. I learned from him to realize the potential in every person and the value of having a positive forward-oriented view of the world. I also would like to thank Prof. Gudrun Klinker for accepting to be my second advisor and providing me with valuable advice to improve my work.

I am grateful to the advice and support given to me by Dr. Asa MacWilliams, Dr. Markus Sauer, and Dr. Hubertus Hohl from Siemens Corporate Technology. The opportunity to talk and work with them, specially the iPraktikum projects with Dr. MacWilliams, gave me great insights into my work.

I am very thankful to my dear colleagues and friends from Chair for Applied Software Engineering, specially Juan Haladjian, Sajjad Taheri, Jan Knobloch, and Jan Ole Johanßen for their warm support and help. I am also very grateful to the help and support from our chair staff Ms. Markel, Ms. Schneider, Ms. Demmel, and Ms. Weber. I would like to thank my dear friends Amin Abouee for very fruitful discussions, and Hamid Momeny and Faraz Zareian for their caring and nice occasional messages.

And most of all I want to express my thanks, gratitude, and love to may dear parents, to my sisters Nazli and Shirin, to Arne, Hasan, Valentin, Leandra, and to my dearest Sara. Without your love and warm support I would have never achieved many things in my life. To Sara, I am also most grateful for her patience and love in the hard times and teaching me to be mindful, take a deep breath and continue.

The responsibility concerning the content in this dissertation is left to me, the author.

Abstract

Authoring systems and end user development toolkits allow end users to develop mixed-reality applications tailored to their specific needs. Many authoring systems are available for augmented reality applications for head-mounted, hand-held, and stationary displays. However, authoring systems for the development of projected display applications are missing. Projected display applications offer advantages when bi-manual interaction and ergonomic convenience are important, in particular in application domains such as preschool education, manual assembly, manual activities like cooking, and medical rehabilitation. Developing augmented reality tabletop systems involves several challenges, in particular a high entrance barrier that prevents end users and experts from non-technical domains, such as education, to experiment with this technology.

This dissertation introduces ISAR, an authoring system for augmented reality tabletops targeting users from non-technical domains. ISAR allows these users to create interactive applications for a tabletop without technical expertise. ISAR consists of an interactive tabletop with a camera-projector setup, a framework and a reference implementation. By hiding the low-level technical complexities, ISAR allows end users to experiment with the development of their own tabletop application in their specific application domain.

ISAR was evaluated in a qualitative usage study with focus on usability and user experience as well as two demonstrations cases. The usage study showed that users without technical background can develop interactive augmented reality applications. The demonstration cases showed that ISAR can be used to create interactive tabletop applications for different domains ranging from manufacturing and medical rehabilitation.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

"Nothing ever becomes real 'til it is experienced."

John Keats

The early 1980's witnessed the introduction of graphical user interfaces based on the WIMP (Window, Icon, Menu, Pointing device) paradigm, which in combination with personal computing are still after more than three decades the dominant paradigm for human-computer interaction. Starting around 1990 new hardware technologies and further miniaturization have opened to door to so-called *post-WIMP* user interfaces [VD97]. The promise of post-WIMP user interfaces is to further naturalize the interaction between human and the computer by decreasing the cognitive barrier that is inevitably imposed by the very existence of a user interface. Mobile computing and multi-touch interfaces are the most present and already established examples of post-WIMP user interfaces.

The evolution of user interfaces from batch processing to post-WIMP follows a trend towards introduction of higher-level abstractions and metaphors that are more familiar to the user [CMK88]. Shells mimic conversations, WIMP interfaces rely on point and select interaction abstractions that are aligned with our mental model of the physical worlds. Tangible user interfaces and interactive tabletops continue this trend by letting users physically interact with digital content [IU97].

The potential of interactive tabletops and tangible user interfaces has been shown in different areas such as education [OF04, ER14], manufacturing [UFG$^+$16], as well as smart homes [ZGN$^+$11]. For example, by providing new forms of interaction, such as direct input and integration of physical and virtual worlds, interactive mixed-reality learning environments support educational settings in which kinesthetic and experiential learning are of importance [ER14]. However, the development of interactive tabletop applications in educational settings is still mostly done by computer science experts and not by teachers [SMMM17].

The advances in hardware and software technologies and the reduced cost of equipment have reached a point that encourages the development of interactive mixed-reality applications by end users . However, their development and adoption still involves several challenges:

- **Levels of Abstraction:** Developing augmented reality interactive tabletops spans different abstraction levels. At the highest level, the end users must understand different technical terminology and concepts, like throw ratio, or frame-rate. The end user must also deal with different device configurations and choice of equipment that are more complex than the familiar desktop PC or handheld mobile devices [Har14]. At the lower level, development of interactive applications for the tabletop requires different technologies such as object recognition, tracking, and rendering.
- **Specifying interactions:** Unlike the familiar WIMP paradigm, interactions in the physical world are much more complex. Development of interactive tabletop applications involves analyzing and defining these interactions for a specific application. Since these interactions cannot be foreseen in advance, changing and adopting them must be easy [WW11]. However, specifying these interactions is a difficult technical challenge [SWMJ10]. End users should be able to define these interactions themselves, and adapt them to their specific domain.
- **Choosing the augmented reality display:** The choice of augmented reality technology has a strong impact on the possible interactions. The advantage of using projected displays on an interactive tabletop is that it supports bi-manual interaction with world objects. Head-mounted displays (HMDs) also offer this advantage, but they require technical knowledge (e.g. battery charge, field of view, resolution) from an essentially non-technical user. Mobile handheld augmented reality, for example with tablets or smart phones, restrict the interaction modes when both hands are required.
- **Limitation in digital content creation:** Tools for content creation by the end user should offer an WYSIWYG experience. However, most frameworks and tools require the creation of digital content for augmented reality applications mostly to be done through programming. Few frameworks address the end users as a separate stakeholder. This problem has been addressed by several high-level authoring tools that allow end users to create their own augmented reality applications [ATH$^+$18]. However, these

tools do not provide authoring capabilities for interactive tabletops that require no programming.

End user development (EUD) environments [NPD13] help to address some of these challenges by empowering the users in creating their own applications tailored to their specific requirements. The necessity of end user development environment for utilization and adoption of new technologies is well stated by Gerhard Fischer [Fis13]: "In a world where change is the norm, EUD is a necessity rather than a luxury; because it is impossible to design artifacts at design time for all the problems that occur at use time."

The diffusion of innovation theory by Rogers [Rog03] describes how new innovations are adopted by a community and provides a theoretical framework for motivating end user development environments. Rogers enumerates five innovation characteristics that influence its rate of adoption: (1) relative advantage or the perceived improvement over current practice, (2) compatibility and consistency with the current practice, (3) how difficult it is to learn, understand and utilize the innovation, (4) trialability or how easy it is to try and experiment with the innovation, and (5) observability or the degree to which the result of the adoption of the innovation is visible to the community. It is clear that providing the users with a tool to create mixed reality interactive applications on their own, allows them to better evaluate these aspects and improves the adoption rate of the technology.

According to Rogers, adapters of an innovation can be grouped into five categories based on the time the adapters start using the innovation. Each of these categories is distinguished by different socio-economic characteristics. The target group we address in this dissertation are innovators and early adapters. This group is characterized by their willingness to take risk and try new innovations as well by their higher socio-economic profile such as level of education, social status, and financial resources. We envision, for example, that ISAR encourages teachers to start developing their own augmented reality and interactive tabletops applications for their students.

1.1 Objectives and Scope

The main objective of this dissertation is to provide end users without technical skills with a solution for creating interactive applications that run on a camera-projector-based tabletop. Such a solution should address the challenges involved in the development of interactive spatial AR applications and hide the technical levels of abstraction from users. Using ISAR end users should be able to create applications for the interactive tabletop

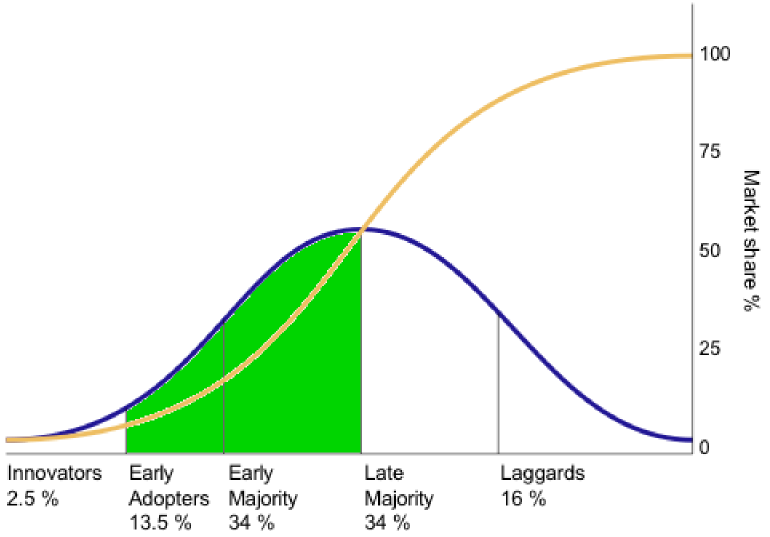


Fig. 1.1 The diffusion of innovation theory [Rog03] describes how usage of new technologies is propagated with the time in the communities. The focus of this dissertation is on early adapters and early majority.

without any programming language skills. The target system should provide the users with the basic support for crating augmented reality applications, including object tracking, object recognition, and registration of digital content on physical objects. Furthermore, the authoring environment should allow users to define interactions with physical objects and digital content associated with these objects.

Application creators (e.g. teachers) are end users that are able to create interactive applications combining physical objects and digital content without the need for programming. They also define the interactions with the tabletop. Application creators don't need to have the technical expertise for creating interactive AR applications: ISAR provides them with an authoring environment for creating their applications. *Application users* (e.g. students) are end users of the created applications. They also do not have to have any technical expertise. ISAR provides the runtime environment for applications users.

In addtion, ISAR supports framework extenders: These are developers with programming experience, but not necessarily experienced in developing AR applications. They extend the ISAR framework, authoring and runtime environment to support new application scenarios.

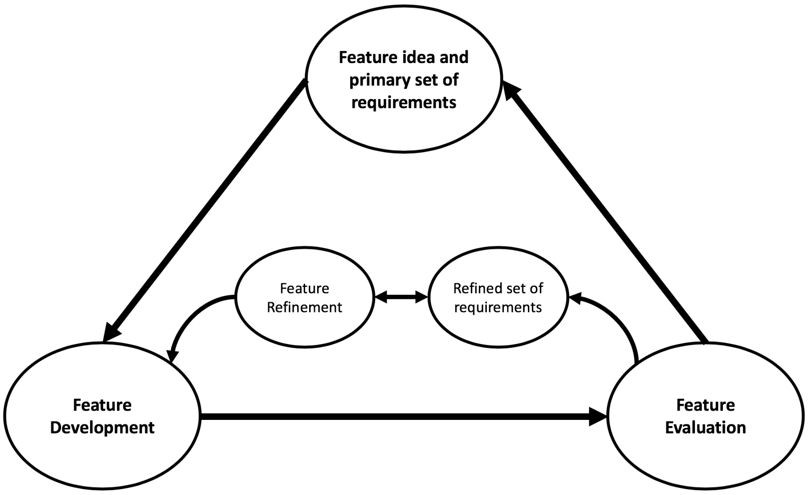

Feature idea and primary set of requirements
Feature Refinement
Refined set of requirements
Feature Development
Feature Evaluation

Fig. 1.2 In a formative research approach the development is guided by iterative evaluation and refinement of the artifacts.

1.2 Research Process

We followed a formative research approach for development of ISAR. In a formative research approach, the research process is guided by repeated evaluation and refinement of the generated artifacts. In contrast, in a summative approach, one would first develop the technology and then evaluate it. Following the formative approach, we developed ISAR in a continuous cycle of implementation, evaluation, and refinements (figure 1.2). Each cycle resulted in a new set of requirements which where again implemented and evaluated.

Beside the formative approach for the development of ISAR, we gained a lot of insights into requirements of ISAR from several other research projects, that were accomplished during the course of this dissertation with industrial partners. Table 1.3 gives a summary of some of these projects. Although the majority of these projects where based on mobile augmented reality, working on them gave us insights into the requirements of an authoring system targeted at end users, most specifically with regard to complexity and cost of the solution, effort for creating content, and defining of user interactions. The main lesson learned from these relevant projects was the need for an end user environment for creating AR applications that hide the involved complexities and allow end users without technical expertise to create AR applications for their own needs. The subject of this dissertation is to address this need by developing ISAR, an authoring system and runtime environment for interactive tabletops.

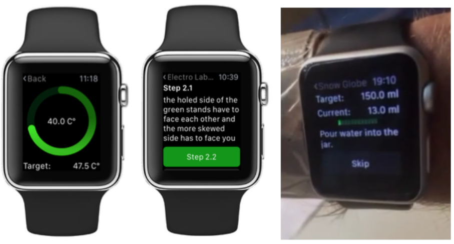
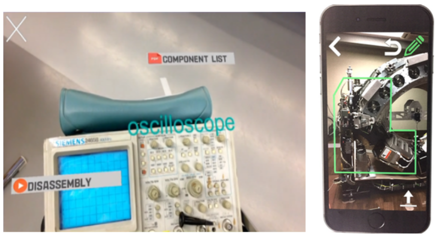
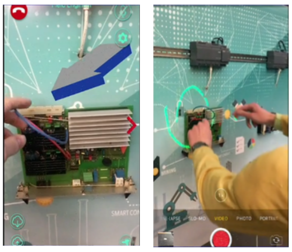
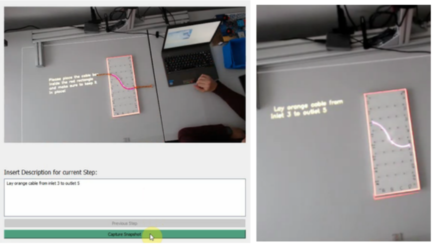
MagicTouch: Guiding users through an industrial workflow using wearable sensors, smart watch, and mobile augmented reality.	 Back 11:18 40.0 C° Target: 47.5 C° Electro Lab... 10:39 Step 2.1 Step 2.2 Target: 150.0 ml Current: 13.0 ml Pour water into the jar. Skip
MindSight: Crowd-sourced augmented reality annotations and labelling of data for object recognition. Image-based archival and retrieval of user generated content and multimedia documents.	
ViewGuide: Remote collaboration with shared scenes and real-time annotations using both mobile and projected augmented reality.	
Alach Demo: Spatial augmented reality using a camera-projector setup for guiding placement of electrical cables in a cable box. Immersive authoring using a programming-by-demonstration approach.	 Insert Description for current Step: Lay orange cable from inlet 3 to outlet 5

Fig. 1.3 Some of the related projects, implemented in course of this dissertation, that gave insights into requirements of ISAR.

ISAR consists of an authoring system for creating the content and a runtime environment that projects the content on the tabletop. We investigated different aspects of the system in different research probes which were developed prior to the development of ISAR:

- Semi-automatic authoring of workflows: Using a depth camera, and predefined tracking areas, this application automatically captured the workflow steps as the expert was performing it. The expert then could add different multimedia content such as text, images, and videos to each step. The resulting content could then be projected on the table to guide the workers performing the workflow. This application however suffered from many shortcomings, most specifically, the need to fine tune the positions of content for each execution, as well as lack of object recognition and tracking.
- Workflow guidance for assembly and cooking: This application was the first version of the runtime environment that projected the content on the table that guided a user through the steps of a workflow. What should be projected at each step, and its position on the table, where predefined in a description file. This application also required a depth camera to track user's interactions, and its main shortcoming was the lack of extensibility and reusability of the content for new scenarios.
- Presentation of multimedia content on the tabletop based on object recognition: This application used object recognition to project different user defined multimedia content such as text, images, audio, and video registered on the physical objects on a table. What should be projected and its position was defined using an environment that showed the topview of the table. The main shortcomings of the this applications were unreliable object recognition and lack of a way to define interactions with the table.

The development of above research probes revealed many challenges and requirements, such as need for simple off-the-shelf equipment, robust object recognition, WYSIWYG authoring, reusability of content, and specification of interactions. We addressed these requirements in the current version of ISAR.

We evaluate ISAR by demonstrating how it can be used to develop applications in different domain (Table 1.1), and how it significantly reduced time and expertise required for developing interactive applications that involve the physical world and run on a tabletop. In a user study we asked the participants to use ISAR to create an application for learning vocabulary. We gave the participants the specification of an application that should help users learn name of tools for a household toolbox. The purpose of this study was to gain insights on how participants use ISAR and evaluate its usability. Furthermore this study should give us indicators on how ISAR can facilitate the development of such interactive applications with regard to development time and effort. We further evaluate how ISAR can be used to create interactive tabletop applications for different domain using two demonstration case studies. In one case study an interactive application is developed for guiding through a workflow for mainboard assembly. In the second case study we demonstrate how ISAR can be used to create an interactive application for hand-eye coordination exercise for post-stroke rehabilitation.

Application	**Domain**	**Description**
Situated learning of vocabulary	Education	This application shows how a teacher can use ISAR to create learning content and students' interactions for situated vocabulary learning [SLR$^+$14]. In the course of a user study we asked participants to create an application that helps users learn the names of tools like screwdriver and pliers.
Guidance through a workflow	Assistance Systems in Manufacturing	This application shows how ISAR can be used to create content that guide users through a workflow. We create an application that guides user on the steps to assemble a computer mainboard.
Hand-eye coordination exercise	Rehabilitation	This application shows how ISAR can be used to create applications with feedback that help users gain motor skills, for example for post-stroke rehabilitation.

Table 1.1 Example applications for evaluation of ISAR.

1.3 Contributions

This dissertation makes the following contributions:

Exploratory review of literature on existing authoring environments for augmented reality and interactive tabletops targeted at end users

Several authoring frameworks and environments have been developed with the goal of facilitating development of augmented reality solution. These authoring solutions range from low-level programming frameworks to high-level content creation tools targeted at end users. We provide a review of the existing literature, with focus on high-level authoring environments for spatial augmented reality and interactive tabletops.

Requirements specification for an authoring system for interactive tabletops
Based on the review of the literature and several research projects we have extracted the requirement for an end user authoring environment for camera-projector based interactive tabletops.

Design of a framework for an authoring system for interactive tabletops
Based on the requirements we have developed the ISAR framework to support the creation and execution of interactive tabletop applications.

Reference implementation of ISAR
We provided a reference implementation of an authoring system for end users to demonstrate how ISAR can be used for the creation of interactive applications.

Evaluation of ISAR
We evaluated ISAR with a qualitative user study of an application for situated vocabulary learning, and two demonstration cases: a workflow guidance for computer mainboard assembly and a hand-eye coordination exercise for motor rehabilitation.

1.4 Organization of the Thesis

This dissertation is organized in six chapters. Chapter 2 discusses the foundations of augmented reality, tangible user interfaces, and interactive tabletops. It also provides a review of related work in high-level authoring systems for augmented reality and interactive tabletops. Chapter 3 discusses the requirements of ISAR in terms of features and requirements for an authoring system for interactive tabletops. It presents several visionary scenarios which are then formalized in a use case model and the specification of functional and non-functional requirements. Chapter 4 introduces the ISAR framework and describes its reference implementation. We discuss design goals and architecture of ISAR and its authoring and execution environments. Chapter 5 describes the evaluation of ISAR with a user study and two demonstration cases. Chapter 6 concludes the dissertation and provides directions for future work.

Chapter 2

Foundations

ISAR is an authoring system for an interactive tabletop based on a camera-projector setup. In this chapter we provide an overview of the related research areas: augmented reality, interactive tabletops and tangible user interfaces, and authoring systems that enable end users to create interactive augmented reality solutions.

2.1 Augmented Reality

Mixed reality is a term used to describe applications that combine the real world with the elements of computer generated virtual worlds. Milgram el al. [MK94] described mixed reality applications along the so called virtuality continuum with the real world at one side and purely virtual worlds, or virtual reality, at the other end (figure 2.1). An augmented reality system overlays digital content onto the real world in a way that from user's perspective the digital information become part of the real world. In this sense an augmented reality application contains primarily the view of the real world enhanced with digital information. In contrast, an augmented virtuality application would be primarily set up in the virtual world with elements of real world, for example body and face of the real users, integrated into the virtual world. The most widely accepted definition of augmented reality, provided by Azuma [Azu97], distinguishes an augmented reality system by the following three characteristics:

- Combining real and virtual: An augmented reality system combines virtual artifacts and digital information with the real world.

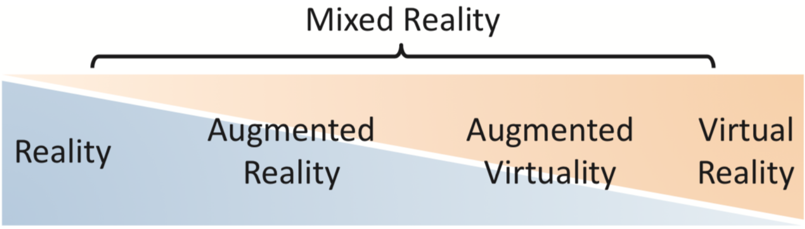


Fig. 2.1 Milgram's virtuality continuum describes mixed-reality applications that integrate virtual elements into the real world, with the completely real and completely virtual environments at the extremes. [SH16].

- Registered in 3D: The digital information must have a precise 3D geometric alignment onto the real world objects.
- Interactive in real-time: The projected digital information (the augmentations) change in real time in response to user's interaction, for example user's view point changes.

Azuma's definition does not limit augmented reality to a specific display technology or even to the visual content. Nevertheless, visual presentation of digital content is the most common form of augmentations.

2.1.1 Method of Augmentation and AR Displays

In contrast to a conventional display, that presents only the computer generated digital content, an AR display needs to combine the view of the real world with the digital content. There are three major methods of augmentation for achieving this combination: optical see-through (OST), video see-through (VST), and projection-based or spatial AR (SAR) (figure 2.2). In see-through displays, the user sees the world through the lens that combines the view of the real world with the augmentations. In an *optical see-through* display a semi-transparent material, such as a half-silvered mirror, is used as the "optical combiner". The light from the real world can pass through this material. At the same time the computer-generated augmentations are reflected on the surface of the display and hence combined with the view of the world. In a *video see-through* display the user does not see the world directly but through a video stream captured by a camera and shown in the display. A "digital combiner" combines the augmentations with the video stream of the world. In both optical and video see-through AR the digital content and the image of the world are combined at the display image plane. In contrast, in *spatial AR* the digital content are projected directly onto the real world objects using a projector instead of a display.

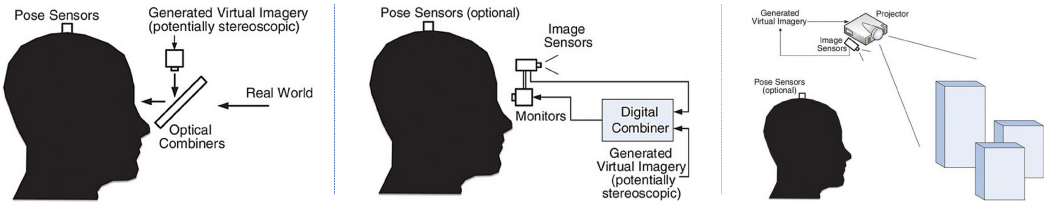


Fig. 2.2 Three main methods of augmentation for combining digital content withe the real world: optical see-through (left), video see-through (middle), and spatial AR (right) [SH16].

Each of the above augmentation methods involve specific advantages and challenges [BCL$^+$15]. The main advantage of the optical see-through displays is the direct view of the real world: the user perceives the real world directly through the optical combiner and not through the video stream as in the case of video see-through displays. The direct view of the world does not suffer from limitation of VST displays, such as camera resolution, lens distortion, and eye displacement. The physical combination of the real world view with the rendered digital content also requires less computational resources compared digital combiner in a VST display. At the same time, this direct view of the world is the source of limitations in OST displays. The spatial relationship between user's eye and the display need to be determined in a calibration step. This calibration is often not accurate and its parameters might change as the AR application is being used. This leads to less accurate registration of digital content onto the real world compared to VST displays, where computer vision techniques can be used for pixel-level accurate registration. Furthermore, because of the direct view of the world maintaining visual consistencies are more challenging in OST displays: achieving the correct occlusion becomes challenging on the the semi-transparent optical combiner and lighting conditions affect the perceived brightness of the digital content more. Another limitation of OST displays over VST displays is the temporal delay between real world view and registration of the rendered augmentations. This delay is caused by tracking and cannot be completely avoided, because rendering of the augmentations depends on the result of the tracking.

Realization of OST displays requires specific materials and parts to realize the optical combiner. In contrast, a video see-through augmented reality system can be realized by any conventional camera and display. Accordingly, VST systems can be more easily implemented on a PC, laptop, smartphone or tablet which eases their adoption. By using computer vision techniques for digital combination of the augmentations with the world view in the video frames, it is possible to achieve pixel accurate registration and better control over visual consistency such as lighting, shadows, and colors, and occlusion. Additionally, since the complete rendered augmented scene is presented to the user through the video frames, there is no perceptible time delay between the world view and registered augmentations. Furthermore, by using wide camera lenses and larger screens it is possible to achieve wider Filed-of-View (FOV) in VST displays compared to OST displays. The main limitation of VST displays is however the indirect view of the world which causes limitations in terms of resolution, distortion, updating delay and eye displacement. Also, because of the digital combination of world view and the augmentations, the VST displays require more computational resources compared to OST displays.

Compared to the common realizations of the VST and OST displays, the projection-based spatial augmented reality (SAR) displays have the advantage of being less obtrusive and more ergonomically suited for certain tasks. Using a SAR display the user usually does not have to wear or hold the display (see below). This advantage can be at the same time a limitation, because in most SAR setups the projector is fixed to a location and hence the AR application is limited to that locations. Of course body-worn, e.g. [MM09], and portable, e.g. [GLC15], SAR setups exist that compensate this limitation. SAR systems also need to overcome the challenges related to projection. The SAR displays require a physical surface that is appropriate for projection, usually a bright, light-diffusing, texture-less surface. Also, the depth of focus (sharpest pixels lie on the focus plane) limits the projections to the nearby objects which causes limitations for application of SAR systems for longer distances in out-door applications. Additionally, because of the projection, the SAR systems are more sensitive to lighting condition and cannot be used in a very bright environment.

Apart from the method of augmentation, the augmented reality displays can also be categorized based on their distance from the eye [BR05] (figure 2.3). Figure 2.4 shows examples of AR displays according to this categorization. Head-attached displays are displays worn by the user, and lie in the head space in terms of distance from the user's eye. The most prominent kind of head space displays are head-mounted displays (HMDs). Although other kinds of head-attached displays such as retinal displays, and head-mounted projectors have also been investigated in research [BCL^{+}15]. Head-mounted displays can be categorized into *optical see-through head-mounted displays* and *video see-through head-mounted displays*. Optical see-through HMDs use an optical combiner that mixes the direct view of the real world with the reflected augmentations that are usually rendered on miniature LCD display. The video see-through HMDs feature cameras for capturing the world view and display it along with the rendered augmentation on the LCD displays. Beside the typical challenges and advantages of the see-thorough augmentation method (see above), a major issue with the HMDs is the trade off between the ergonomic design and visual properties of the display. A HMD must be lightweight, adjustable to different head sizes, and suitable for longer usage. In the case

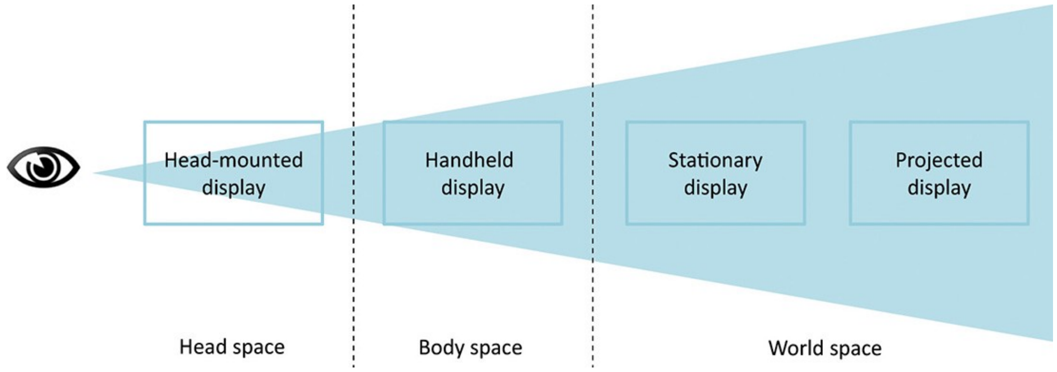


Fig. 2.3 Different AR display categories based on the distance from user's eye. [SH16].

of an stand-alone (not tethered) HMD, the battery life must also support reasonable period of usage. Tracking, field-of-view, resolution, and visual consistency are other issues that need to be addressed on a usable HMD. All these issues have lead to the fact that augmented reality HMDs are less available and adopted to the general consumers and are mostly used in large enterprises and still in research and development settings [ME19, KNG16][1].

Handheld AR displays usually offer video see-through AR implemented on a consumer smartphone and tablet that user holds in the hand. The back camera of the handheld device captures the view of the world which is presented to the user along with the rendered augmentations through the device screen. The main advantage of the handheld displays is the large-scale availability of smartphones and tables to the end users. These devices mostly feature considerable computing power as well as different sensors that can be used for tracking. This availability has boosted the development of handheld AR applications and has positively influenced the adoption of augmented reality into different applications domains [PROM18, SH16, BCL^{+}15]. The major issue with handheld AR displays is obtrusiveness and the need to hold the device with at least one hand. This limits the usage of handheld AR systems for the activities that require both hands. Furthermore, holding the device might lead to unstable viewpoint which can be limitation for the concentration and accurate applications.

Moving further form the user's eyes, are world space or spatial displays. These displays are not worn or held by the user, but are either stationary in the world, or projected directly onto the objects. Stationary displays can be used for optical see-through or video see-through systems. In video see-through systems, a camera records the view of the world, which is

[1] As an example, Microsoft Hololens 2, one of the latest and most advanced developments in optical see-through augmented reality HMDs, weighs more than 500 gram and offers a horizontal FOV of 43°and vertical FOV of 29°and is priced 3500$ (https://www.microsoft.com/en-us/hololens/hardware, last visited January 2020)

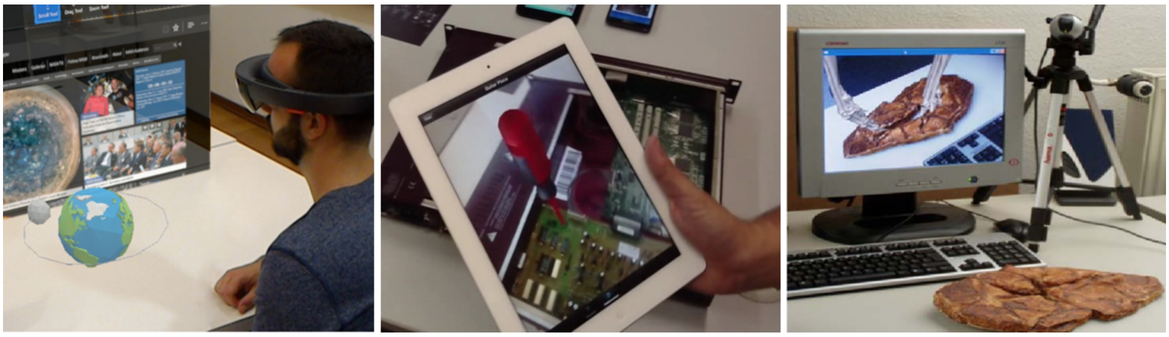


Fig. 2.4 Examples of AR displays according to categorization based on distance from the user's eye: (left) Microsoft Hololesn, a head-mounted optical see-through display [SSDM17], (middle) handheld display [GMOF13], (right) stationary display for screen-based video see-through AR [BGW+02].

the then displayed along with the rendered augmentations. Stationary displays can be used in several different setups such as desktop displays (figure 2.4, right), virtual mirrors (e.g. [BKBN12]), and window and portal displays that change the rendered content according to the tracked user's viewpoint (e.g. [ZSDS13]). Optical see-through stationary displays may be implemented similar to the optical see-through HMDs using a large enough semi-transparent optical combiner (e.g. [BF02]).

Projected displays are another category of world space displays. In contrast to optical see-through and video see-through displays, in projected displays the augmentation are directly projected onto the surfaces and objects of the real environment. Spatial projected displays are not limited to one fixed projector and approaches using head-mounted projectors (e.g. [KS03]) and handheld mobile projectors (e.g. [RVBB+06]) have been proposed. Spatial projected displays with spatially fixed projectors have been implemented in different variations. Raskar et al. [RWLB01] used computer vision techniques and developed methods for precise 3D illumination and projection of animations on non-trivial complex 3D objects (figure 2.5, left). This approach is called *shader lamps*. Dynamic shader lamps [BRF01] combines this idea with tracking the objects in the environment to allow dynamic projection of content, colors, and textures onto moving objects (figure 2.5, middle). Augmented surfaces [RS99] are a broader category of using projected displays to augment mostly planar objects and surfaces such as desktops. These systems mostly combine 2D augmentations of physical objects with tangible interaction. ISAR is a an example of such interactive surface-based augmented reality and we provide a more in-depth review of these systems in the next section.

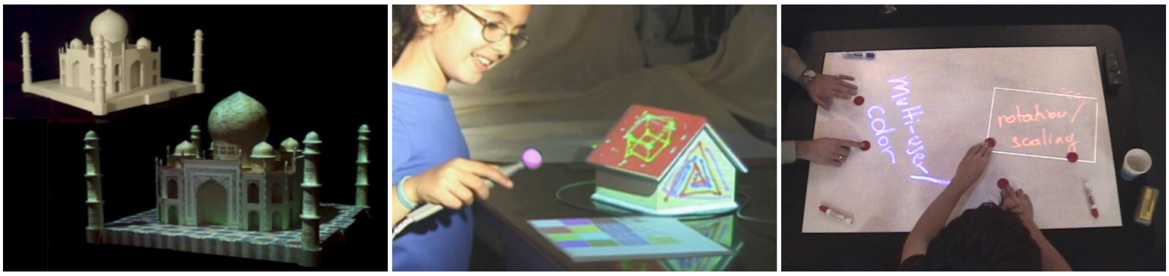


Fig. 2.5 Examples of projected spatial AR displays: (left) illuminating 3D non-planar objects with shader lamps [RWLB01], (middle) painting by light using dynamic shader lamps [BRF01], (right) the Magic Table, an example of an augmented surface [BGW$^+$02].

2.1.2 Components of an AR System

An augmented reality application overlays digital content onto the view of the real world. The augmentation happens in real time in response to user's interactions and the AR system provides precise 3D alignment of digital content. To achieve these goals, an AR system must have at least three components, tracking, registration, and visualization, that work together in a real time feedback loop (figure 2.6).

Prices 3D alignment of the augmentation onto the real world objects requires knowing the 3D position and orientation of these objects relative to the view point. The user can interact with an AR system in several different ways, for example by changing the view point through moving the camera, head, or eyes, or even moving the body in the environment. Manipulating and moving the objects in the environment is also a possible way of interaction. The change of the view point or the real world scene is captured using the *tracking* component. The tracking component determines the pose, i.e. location and orientation, of the real world objects relative to the view point. Depending on the setup of the AR system, such as method of augmentation and display type, the pose of the camera or the pose of the objects in the environment might be tracked. The estimated pose from tracking is used by *registration* component for 3D alignment of virtual objects onto real objects. Finally, the *visualization* component renders and visualizes the augmented scene. The visualization can be situated, meaning it retrieves and presents the information in a context-aware approach [WF09]. Figure 2.6 also shows a fourth component: the spatial model. The spatial model is persisted source of information about the virtual augmentations as well as about the model of world. The content of augmentations, i.e. the virtual information and artifacts, are stored in the virtual model. The model of the real world serves as the reference for tracking. The registration component registers the digital content retrieved from virtual model onto the the model of the real world using the tracking information.

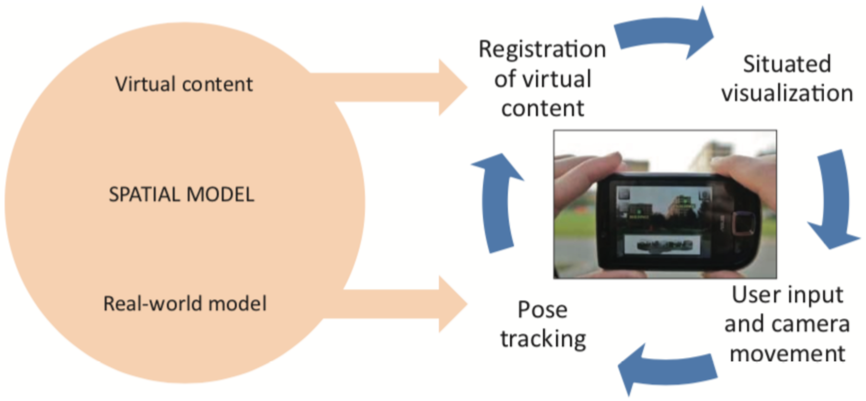


Fig. 2.6 Main components of an augmented reality system and how they contribute to the human-computer feedback loop. The integration of virtual content into the view of the world should happen at real time, meaning the user should preferably not perceive any delays in rendering of the augmentation onto the view of world. [SH16].

Tracking

Registration of digital content onto the real world objects requires knowing the relative position and orientation of the objects with respect to the view point. The position and orientation of the objects is also called pose. Different entities can be tracked in an AR system: user's body, eyes, head, pose of the camera, display, and the physical objects in the world. To make real time registration of digital content possible, tracking continuously updates the information about the pose of the objects during runtime of an AR application.

The tracking provides pose information relative to a reference coordinate system, and depending on the setup of the AR system different transformations are needed to align virtual content onto the view of the real world. A concept related to tracking is *calibration*. Tracking continuously provides information about dynamic spatial relationships in the runtime. Calibration describes the spatial relationships that does not change dynamically and continuously. Calibration in general determines the static relationship between measurements of a sensor and a known scale. In AR, calibration determines static parameters and transformations between components of the AR system that do not change dynamically, such as internal camera parameters. Calibration is usually performed at the start of an application or at discrete intervals.

Depending on the AR application, several different tracking approaches exist. These approaches can be characterized based on different properties such as physical phenomena, spatial arrangement of sensors, degree of freedom, and so on. Examples include stationary mechanical or electromagnetic tracking systems, GPS tracking and tracking based on mobile

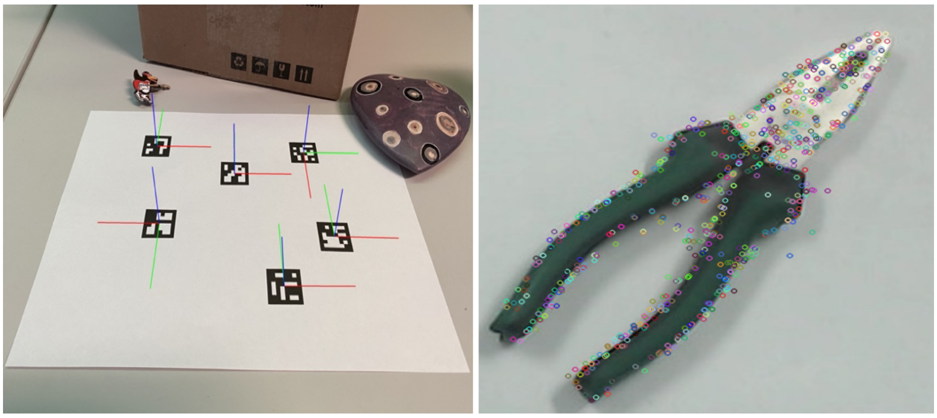


Fig. 2.7 ArUco markers [RRMSMC18] with their corresponding coordinate systems rendered on the image[2](Left). Keypoints for natural feature tracking extracted using AKAZE algorithm [AS11] and drawn on the image (Right).

motion sensors such as gyroscope and accelerometer, and optical tracking. In this section we mention only briefly two optical approaches related to ISAR, namely marker-based and natural feature tracking. For a detailed overview of different tracking approaches refer to [SH16].

Optical tracking approaches use computer vision techniques to estimate the pose of the camera or objects from captured images. In general determining the pose requires solving correspondence problems, in which a geometric transformation is found that maps pairs of corresponding points in two different coordinate systems. This transformation yields to pose of the camera or object. In case of optical tracking the correspondence points are extracted from the features of the environment such as 3D points in the environment and their projected point on the camera image. These features can be natural or artificially added to the environment. The artificial features added to the environment are called *markers* or *fiducials* and tracking based on them is called *marker-based tracking*. The natural features, on the other hand, are extracted from the environment itself, and tracking based on them is called *natural feature tracking* or marker-less tracking.

Tracking using makers is usually faster, more robust, and more accurate compared to natural feature tracking. Markers are simple patterns designed in a way that makes their detection easy and reliable in the image. They usually have distinct features with high contrast for better detection in different lighting condition. The marker pattern can contain an identification code too. Figure 2.7 (left) shows six ArUco [RRMSMC18] markers printed on a paper and their corresponding coordinate systems are augmented on the image based on their

[2] Image from https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html

orientation. In this case, the pose (location and orientation) of the camera with regard to the marker is obtained by detection of the four marker corners and solving the correspondence problem as a homogrphy [SH16]. Each marker also has a unique pattern coded that allows detecting its id. Marker-based tracking has the advantage of being robust and fast. However its main disadvantage is that the markers must be attached to the environment and tracking targets in advance. It sometimes may not be practical to require the users to instrument the environment before they can use the AR application. In this case natural feature tracking should be used.

Natural feature tracking relies on distinctive features that already exist in the image of the environment. These distinctive features can be corners, edges, color and contrast changes, and textures. These points of interest in the image of an object are called *keypoints* (figure 2.7). There are several methods for extracting the keypoints in an image that rely on mathematical description of the keypoints based on pixel values in the image. For determining the pose of the camera and detection of the objects, the keypoints are extracted in the camera image. These keypoint are then compared with features of the model of the object that has been created and saved in a database in advance. This process is called *feature matching*. Natural feature tracking has the advantage that it does not require instrumeting the environment and tracking targets in advance with artificial markers. However, it requires better image quality and more computational resources. Natural feature tracking is also sometimes not reliable and robust, for example when the objects and the environment do not provide enough texture to extract unique and distinct feature points.

Regardless of the method (marker-based or natural feature tracking), tracking requires a model of the tracked objects or the environment as the reference model. When this model is known in advance, this is called model-based tracking: the features are extracted from the camera image and compared to the known model to obtain the pose with reference to that model. If the model is not available in advance, it must be build on the fly, just before tracking starts. This initial model is then usually updated and enhanced as the tracking continues. With model-free tracking, virtual objects cannot be pre-registered to the model (because it is not know a priori) and the tracking is relative to the starting point.

2.2 Tangible User Interfaces and Interactive Tabletops

The promise of augmented reality is to seamlessly combine the real world surrounding the user with digital information. In fact, augmented reality is one of the many different new trends in human-computer interaction that can be described in the larger analytical frameworks of tangible computing [Dou04] and reality-based interaction [JGH$^+$08]. Augmented reality, ubiquitous computing, tangible user interfaces, and interactive surfaces and spaces are among these new trends in HCI. Dourish [Dou04] distinguishes three main themes in tangible computing HCI trends: distribution of computation across many devices in the environment; augmenting everyday familiar environments with digital information; and enabling the users to interact with these systems through physical activities and objects.

Common to all these trends is distancing from established desktop computing interaction model with clearly defined input and output devices. Instead the users can interact with the system by utilizing a multitude of devices, real objects, and physical activities. Also, the sequence of interaction in no more exactly defined and the interaction happens ad-hoc. The physical environment often offers affordances [Nor88] that enables user to interact with the system based on his knowledge and skill of using everyday objects.

Reality-based interaction [JGH$^+$08] describes how the existing knowledge and skill of the users is utilized to enable “real-world” interaction with the computer systems. These interactions rely on different level of skills and awareness about the physical objects, user’s body, the environment, and the social context. In the following sections we provide a brief overview of foundations and representational works in the areas of Tangible User Interfaces (TUIs) and Interactive Tabletops that are of relevance to ISAR.

2.2.1 Tangible User Interfaces

Building upon pioneering previous work, such as DigitalDesk by Wellner [Wel91] and *graspable interfaces* by Fitzmourice et al. [FIB95], Ishii and Ullmer proposed a new paradigm for human computer interaction called *tangible user interfaces (TUIs)* [IU97]. Tangible user interfaces make it possible to interact with digital information by using physical objects called tangibles. Physical objects and the environment act as both representation and a means to control (manipulation) of digital information. In this way, through coupling of digital information with physical objects, the bits (digital information) become literally tangible, hence the name *tangible bits* by Ishii and Ullmer [IU97].

Ullmer and Ishii [UI00] analyzed different implementations of tangible interfaces and provided a framework for describing the tangible interaction. Contrasting it to the familiar model-view-controller paradigm [KP$^+$88] in designing desktop graphical user interfaces, they argued that MVC highlights the clear separation between the input (control) and output (view). In contrast, the physical representation of a tangible interface embodies both the view and the control. They formalized this new paradigm of interaction in the MCRpd (model-control-representation), which takes the concepts of model and control from MVC but devises the view into two representations: the physical representation (Rp) and the digital representation (Rd). This model also emphasizes the embodiment of the control in the physical representation, that is the tangible is at the same time both the representation of the information and the means to manipulate it. This new paradigm is also called MCRti in other publications, in which tangible (Rt) and intangible (Ri) representations are used instead of physical and digital representations. Leaning on this model, Ullmer and Ishii count four main characteristics for TUIs:

- Computational coupling of physical representation (Rp) and the underlying digital information (the model)
- Embodiment of control in the physical representation: the user can manipulate the model by naturally interacting with the physical representation
- Perceptual coupling between physical representation and the mediated digital representation
- Embodiment of the aspects of the digital state of the system in the physical state of the tangibles

With the increasing number of works based on the idea of TUIs, several classification frameworks are suggested that aim to analyze and categorize different aspects the TUI systems. Ulmer et al. distinguish three main directions in TUI systems (figure 2.8):

- Interactive Surfaces: Interactive surfaces are a class of TUIs in which tangibles are placed and manipulated on an augmented planar surface . Interactive surfaces are usually realized using camera-projector systems and the digital information are projected on the surface from top or below. The application is controlled by tracking different aspects of the tangibles on the surface, such as presence, identity, and spatial configuration and arrangement. Urp [UI99] is a typical example of interactive surface TUI, in which small tangible models of the buildings are used to control simulations projected on the table (figure 2.8 left). Interactive tangible tabletops are subject of this thesis and we will cover them more in-depth in the next section.
- Constructive Assembly: Constructive assembly TUIs are systems based on modular building blocks that are connected together to create an application. The building blocks are usually instrumented with electronic circuitry and by connecting them together in

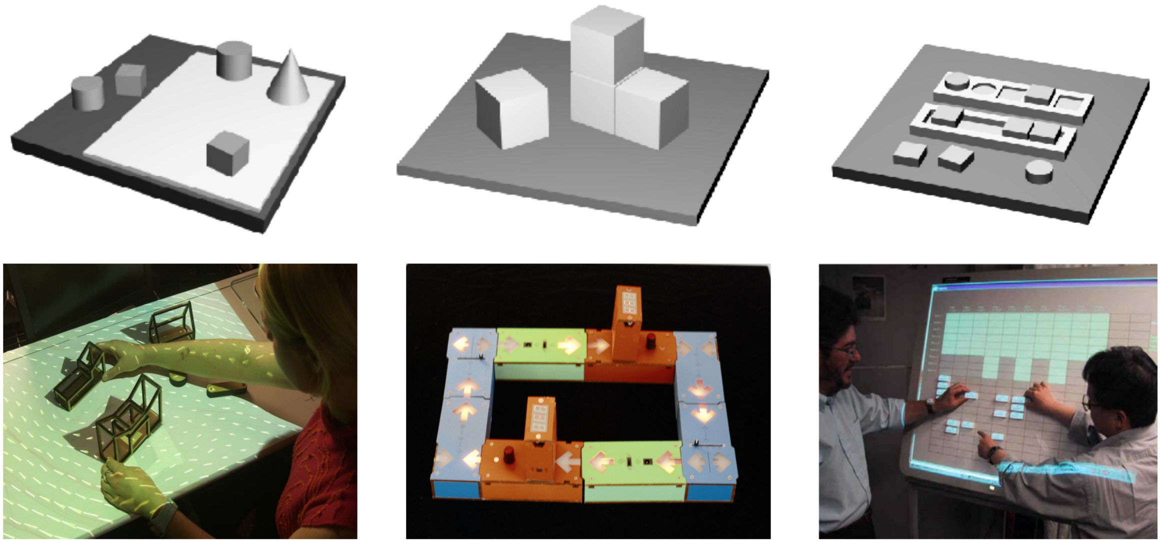


Fig. 2.8 Schematic representation and examples of different types of tangible user interfaces according to Ullmer et. al [UIJ05]: (left) Interactive surfaces, Urp [UI99]; (middle) Constructive assembly, FlowBlocks [ZAR05]; (right) Token+Constraint, SenseBoard [JIPP02]

the specified way the computational functionality is achieved. FlowBlocks [ZAR05] is an example of constructive assembly TUI designed for teaching mathematical concepts like counting and computer-science concepts like loops to children (figure 2.8 middle).

- Token+Constraint: Token+Constraint TUIs use physical constraints to guide mechanical movement of tangible tokens. Tokens are tangible objects that represent digital information and physical constraints are used to associate the movement and arrangement of these tokens to specific computational functions. According to Ullmer and Ishii, Token+Constraints are best suited to interact with abstract computational functions and digital information that do not have an inherent physical representation as in the case of tangibles on interactive surfaces. SenseBoard is an example of token+constraint TUI in which a projected grid constraints the possible placement of small plastic tokens. By placing tokens in specific positions in the grid, a distinct computational functionality associated with the identity of the token is triggered (figure 2.8 right).

Ullmer and Ishii also considered the approaches for coupling digital information and computation with the physical objects, what they call binding. In a system with static binding the coupling of tangibles and underlying digital information is defined in advance by the designer and cannot change during the runtime of the TUI application. In dynamic binding the user can couple tangibles and underlying information himeslf by using the tangible user interface during the runtime of the application. This approach is similar to *immersive authoring* of augmented reality [LKB05] that we will discuss later. In a similar approach for classifying TUIs based on the connection between physical tangible and digital information, Holmquist et al. [HRL99] suggest three types of tangible objects: containers, tokens, and tools. Containers are generic symbolic objects that can represent any kind of digital information. For example an augmented reality marker can act as a tangible container of information. A token is a tangible objects whose form and appearance has representational significance for the digital information it represents. For examples the objects that represent buildings in Urp [UI99]. Tools are tangible objects used for controlling the application and manipulation of the information and representing the computational functionalities.

Another classification framework suggested by Fishkin [Fis04] addresses the conceptual mapping between physical and digital in TUIs. Considering the aspects of human computer interaction and user experience, this classification categorizes TUI solutions across two axes: metaphor and embodiment. The metaphor describes the degree of representational mapping between user actions and their effects in the system and how directly the response of the system can be interpreted. The metaphor aspect of a TUI can range from none, for no connection to a real-world entity (e.g. a command-line interface), to noun, for similarity between shape and resemblance of the objects with a real-world entity, to verb, for when the object mediates the manipulation of the information, to full, in which the user has the impression of direct immediate connection between physical and digital objects. The embodiment axis describes the connection between input and the output in a TUI. The embodiment aspect of a TUI can range from full, in which the input and the output are not distinguishable, to nearby, where the output happens in the vicinity of the input object, to environment, where the output is at the same spatial environment as the user but not directly closed to the input, and to distant, in which the input and output are spatially disconnected, e.g. it happens at separate rooms.

Implementations

Several implementation strategies have been utilized to create TUIs depending on the requirements of the application. These requirements for TUIs applications can be countless, nevertheless there are basic functional requirements that are common in a multitude of applications. These functional requirements can be categorized into several aspects such as detection of presence and spatial configuration of the tangibles, robustness, reliability, and realtime performance of the application, cost, aesthetics, and effort needed to setup and prepare the system for use. Major implementation strategies utilized can be grouped into systems utilizing RFID technology, systems based on microcontrollers, sensors, and actuators, and systems that utilize computer vision techniques to fulfill the functional requirements of the TUI [SH^{+}10].

RFID-based TUIs usually are based on detection of the presence and identity of the objects. Although the cost of their implementation is moderate, RFID-based system need configuration of RIFID reader and the user needs to equip each tangible with the corresponding RFID tags. The systems based on micorocontrollers, sensors, and actuators are more versatile in terms of different features of the physical world that can be detected for realization of the TUI. Based on the sensors utilized, these systems can detect presence and mechanical configuration of the objects and many different physical properties of the environment such as light intensity, humidity, temperature, acceleration, and proximity. However these system are usually costly and complex to setup by end users. Vision-based systems are based on detection of the tangibles using computer vision techniques. Theses system are usually more versatile than RFID-based TUIs in terms of what physical properties of the application can be detected, for example identity, presence, spatial configuration, shape and color of the tangibles. Compared to the TUIs based on microcontrollers, theses systems are less costly and easier to setup by the users. TUIs based on computer vision techniques usually consist of camera-projector setups, in which the input and user interactions are detected by the camera and digital representations and response of the system are projected on planar surfaces. Detection of the objects is often based on fiducial markers and sometimes using markerless natural feature tracking.

Applications

Tangible user interfaces offer many advantages for applications in which situated physical interaction with world is of importance [MO12]. Familiarity of the physical objects and their affordances, reduces the barriers for interaction and supports collaboration [SJZD10]. TUIs also offer the advantage of direct two-handed interaction multiplexed through the space. Two-handed interaction allows parallel performance of complex spatial interaction in a natural way without imposed cognitive load of an extra user interface [KHT06]. Several applications across different domains, such as design, architecture, education, entertainment, and information visualization have utilized the advantages of TUIs. Here we provide a brief review of some of these application domains with examples. Survey studies such as [SH^{+}10], [UI00], and [MWS12] provide a more in-depth and comprehensive overview of the field.

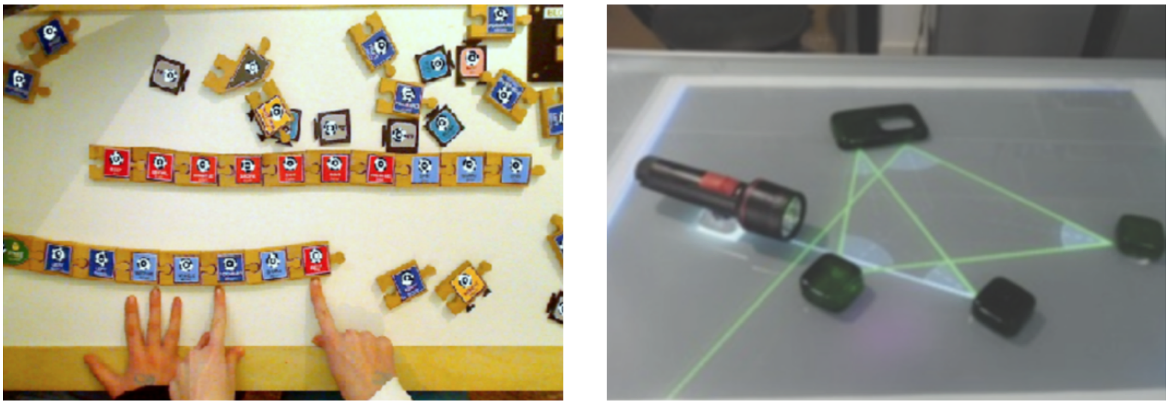


Fig. 2.9 Examples of TUIs for educational applications: (left) Tern [HSCJ09] for teaching programming; (right) An interactive tabletop by Price et al. [PFSR09] for teaching optics.

Educational applications are one of the main application domains of TUIs [SH+10]. Relying on educational theories such as constructivism [Ack01] and hands-on learning [Hei91], these TUI applications take advantage of aspects of tangible interaction such as multiple external representations [Ain99] and relationship between physicality and cognition [Mar07]. Several implementation approaches have been utilized for TUIs for education. For example Smart Blocks by Girouard et al. [GSH+07] is a construction kit designed for teaching mathematical concepts such as calculation of volume and area by building 3D physical models; Using Tern by Horn et al. [HSCJ09] users can create computer program by connecting physical wooden blocks that represent the control flow (figure 2.9 left). Many systems also use tangible interaction on interactive surfaces. For example Price et al. [PFSR09] created a back-projected interactive tabletop that used tangible objects for teaching concepts of optics (figure 2.9 right); TinkerLogistics application by Zufferey et al. [ZJLD09] uses an interactive tabletop with tangible models of warehouse shelves to teach concepts related to logistics by simulating the construction of the warehouse.

Design, problem solving, simulations, and information visualization are other application domains for which several TUIs have been implemented. Aspects of TUIs that lend themselves well to applications in these domains are epistemic actions, physical constraints, as well as tangible representations of the problem [SH+10]. Manipulation of objects can be considered as either epistemic (exploratory) or pragmatic (performatory) actions [KIR95]. Epistemic actions are performed in order to gain insights into the problem and discover information about the context of the task. Tangibles support epistemic actions by allowing user manipulate physical representations that are coupled to computations and more information-rich versatile digital representations. Physical constraint in TUIs reduces the cognitive load imposed by interaction with the computational interface by limiting the space of possible interactions and rules. Finally, the tangible physical representation supports solving problems that are strongly related to spatial and geometric arrangement of objects in the real-world [KM08]. In the following we briefly refer to some illustrative examples.

Urp [UI99], ColorTable [MPW08], and Illuminating Clay [Ish08] are examples of TUIs used for architecture and urban planning tasks. All these systems feature tangible models of the environment, e.g. buildings in Urp or landscapes in Illuminating Clay, that are coupled with the simulations and projection of the information. Collaborative work is also an important aspect addressed in theses systems. Tangible Query Interfaces by Ullmer et al. [UIJ05] are token+constraint TUIs for visualizing information stored in a database. The parameters for queries to the database are controlled by the placement and spatial relationship of tokens. Similarly in GeoTUI [CRR08] tangibles are used to control presentation geological information projected on the surface and in Senseboard [JIPP02] or the system by Edge and Black [EB09] tangibles are used to visualize and plan information organization and office activities respectively.

2.2.2 Interactive Tabletops

Interactive tabletop are a class of interactive surfaces in which the digital representations are displayed on a horizontal or tilted surfaces. This horizontal surface often acts as both the input and output of the system, and supports quite different interactions compared to vertical interactive surfaces [DE11].

Different approaches and interaction forms have been utilized for realization of interactive tabletops. Kunz and Fjeld [KF10] provide a classification of interactive tabletops across dimensions of type of interaction, tracking and identification of user interaction and objects, and displays. Interaction with an interactive tabletop can be based on natural hand interaction, devices such as mouse and stylus, or physical objects. The tracking and identification of interaction and objects can be optical or based on electrical signals or acoustic tracking. The displays can be projection based or using integrated LCD displays. The projection can be either from top of the surface or below it. Top projection systems are usually easier to setup, since they only need hanging the projector on top of any surface. In this section we provide a brief review of setup and interaction approaches and important works of interactive tabletops. The focus will be on projection based displays and natural hand and object based interactions, since these systems are the nearest to ISAR.

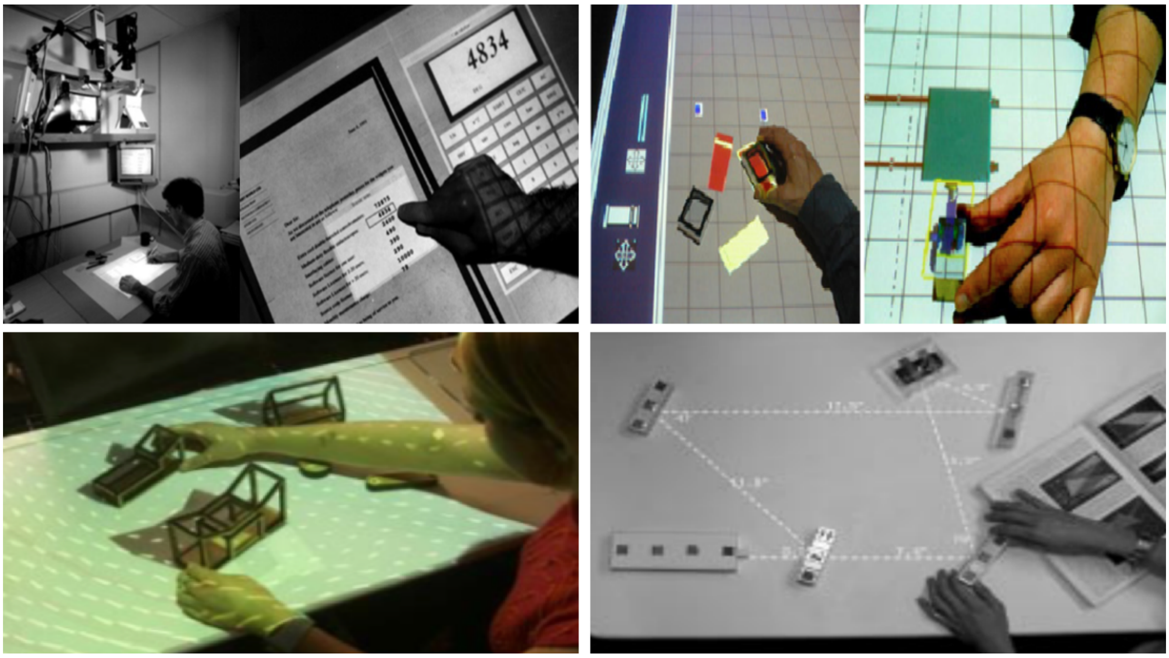


Fig. 2.10 Early works in the area of interactive tabletops: Digital Desk [Wel91] an interactive tabletop interface with touch interaction (top left); BUILD-IT [FBR97] support design tasks by tangible interaction on a projected tabletop (top right); Two applications,Urp and Illuminating Light, from Luminous Room project [UUI99] (bottom)

One of the earliest pioneering works on tabletop interfaces is the Digital Desk by Wellner [Wel91] (figure 2.10 top left). Wellner's work was motivated by the contrast between richness of interaction possibilities with physical paper versus the richness of the computational tasks possible using digital documents. Digital Desk aimed at filling this gap between physical and digital world by combining the physical interaction with digital documents. The system was an office desk equipped with a projector, cameras and sensors. The system tracked finger interaction and recognized content of paper documents using OCR. Digital documents and applications were projected on the surface of the desk. Using the example of a calculator application, in which the user could input numbers by simply pointing at them on the physical paper, Wellner demonstrated seamless blending of the physical-digital interaction.

Inspired by the possibilities of interaction with physical objects, metaDesk by Ullmer et al. [UI97] was designed to investigate tangible interaction with digital projected information. The system displayed information using a back and top projected interactive tabletop with additional mounted LCD screen called active lens. Tangible objects such as a building models or a physical magnifying glass, called passive lens, were used to control the presentation of the digital information. The tracking of the objects and the lenses was done using vision-

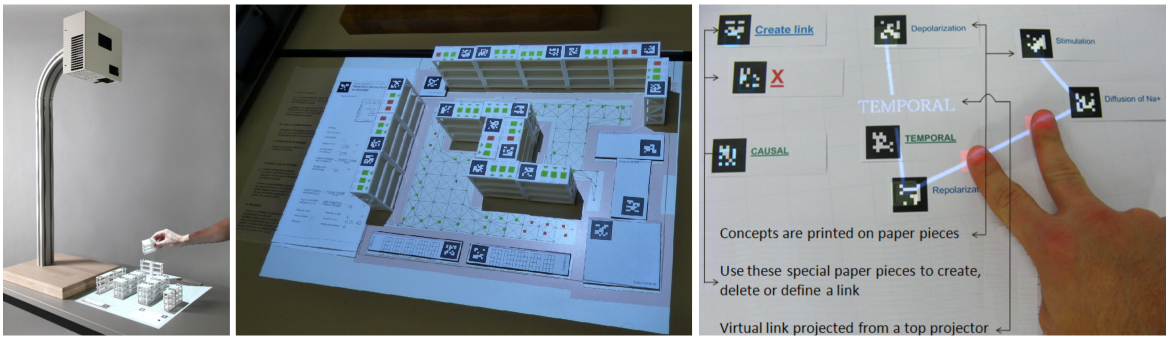

Create link
X
CAUSAL
Depolarization
TEMPORAL
TEMPORAL
Repolarizat
Stimulation
Diffusion of Na+
Concepts are printed on paper pieces
Use these special paper pieces to create, delete or define a link
Virtual link projected from a top projector

Fig. 2.11 TinkerLamps [ZJLD09] and two of its applications: Warehouse logistics simulation and training [SJZD10] and collaborative creation of concept maps [DLKD09]

based infrared light and electromagnetic sensors. This pioneering work was the precursor to the idea of tangible bits and TUIs in general [IU97].

Similar to metaDesk, BUILD-IT by Fjeld et al. [FBR97] was designed to allow physical interaction with digital information (figure 2.10 top right). The system consists of top mounted camera and projector. The interaction with digital information is through manipulation of physical tokens called bricks. The physical appearance of the bricks do not carry any meaning and they can dynamically be associated with virtual objects by putting them on the projected representation of the digital object. They can be detached from virtual objects by hiding the brick with the hand. An IR camera is used to track the bricks on the table.

Another pioneering work in the area of tangible interaction with interactive surfaces is Luminous Room by Underkoffler et al [UUI99] (figure 2.10 bottom). The ultimate goal of Luminous Room project was pervasive projection of information on any surface combined with tangible interaction. They introduced the concept of I/O Bulbs, as integrated camera projector units, that could project information and tracked user interaction and physical objects in any environment. Although not completely built due to technical challenges, they showed the feasibility of this concept through a series of applications. For example in Urp [UI99] the setup was used to support urban planning tasks by simulating shadows, traffics and in Illuminating Light [UI98] they used it for optics simulations. Notable in Luminous Room applications is the direct correspondence between the physical representation of the tangible object and its underlying digital abstraction, as opposed to the symbolic correspondence seen for example in Build IT.

TinkerLamp, by Zufferey et al. [ZJLD09] and Do-Len [Do12] is a camera-projector setup in a compact form factor developed for research in educational applications of interactive tangible tabletops (figure 2.11). Several iterations of TinkerLamps have been developed in the research group Computer-Human Interaction in Learning and Instruction (CHILI) of EPFL university. The system is a top projection projector-camera system in portable form factor, with touch detection, paper interfaces [Do12], and marker-based object detection and tracking, and gaze tracking. TinkerLamps has been developed as a platform for research in educational applications of interactive tabletops in real classroom settings. To this purpose several applications have been developed using this platform including a mind map application [DLKD09], an application for training and simulation of logistics skills and planning of warehouse activities [SJZD10], and applications for training 3D spatial skills of carpentry apprentices [CDZOD15]. In the case of logistics application, the parameters of the simulation can be controlled using paper interfaces [ZJLD09], and various metrics of the simulation are projected on the tabletop for students to reflect about the effect of their decisions.

reacTable [JGAK07] is another influential interactive tabletop system, designed as a music instrument for creating synthesis music. The system consists of a back projected interactive tabletop and tokens equipped with fiducial markers. Each token type has a specific function such as sound generators, filters, and control of sound parameters. The tokens are detected using reacTIVision [KB07] computer vision framework and the state of the tangibles is transmitted to the music software using TUIO [Kal09] protocol. In fact reacTable was the first application that utilized reacTIVision and TUIO. Following reacTable, a large number of tangible tabletop applications have been developed utilizing racTIVision and TUIO, for example for educational applications (e.g. [DRSG18] for teaching concepts of artificial intelligence, [BDOB13] for teaching mathematical concepts related to music, [TBB10] for teaching engineering design, and [SWBP13] for teaching neuroscience concepts).

2.3 Authoring Systems

All projects introduced in previous section are created on top of toolkits and platforms that facilitate creation of interactive tangible tabletop application. These toolkits and platforms provide different low-level abstractions, functionalities, and features, such as object detection and tracking, that are used to create higher-level application for a specific domain. Nevertheless, the development of the applications still require intensive knowledge of technical aspects of the underlying toolkit and the applications are often developed by the creators of those toolkits themselves (who are experts in computer science and programming). With this aspect in mind, we follow our discussion in this section about end user development and systems that support creation of interactive applications by rather non-technical end users.

End user development is the approach to empower users, who are non-professional software developers, in modifying and creating digital artifacts themselves according to their own needs and requirements [LPKW06]. Continuous and fast paced change of the requirements makes often the initial design obsolete, so that it does not meet the requirements of the end users anymore. Moreover, no matter how accurate and intensive the requirements of a software are elicited, developer often cannot know the requirements of a domain and its foundations as good as domain experts. End user developments empowers users, who are possibly experts of their own domains but not experts in programming, to express themselves and create their own digital artifacts independent of computer science professionals [Fis13]. This flexibility and independence has great implications on how users experience usage of digital artifacts in their daily practice [Fis13].

The usefulness of an EUD can be analyzed from perspective of what Meyer et al. [MHP00] call *threshold* and *ceiling*. The threshold of a development toolkit (be it a software framework or a high level end user development environment) describes how hard it is for the user to get started and create useful artifacts with it. The ceiling on the other hand describes the versatility and flexibility of the toolkit and how much can be done with it. Obviously, an EUD targeted at domain experts who don't have any experience in programming and possibly are not at all interested in it, must offer the right balance between threshold and ceiling. It must be possible for the end user to create artifacts that he perceives useful, at the same time getting started with the system and using it should be easy enough so that the end user does not feel overwhelmed. In addition to that, an EUD must also consider the environment and hardware possibilities of the end user. An EUD that requires expensive hardware and sophisticated hardware would not be easily adopted by the end users.

Development of EUD systems involves a multitude of challenges and requirements, such as flexibility and modifiability, ease of learning and usability, and providing the end users with environments to test the artifacts they have created [BCFP19]. It is also important to consider who is the *end-user*: the person who uses the EUD environment to create artifacts for his own use, or for example a teacher who uses EUD environment to create artifacts that will be used not by himself, but by his students [TSM13]. An EUD, targeting non-programmer domain experts, must provide high-level abstractions and enough modularity to allow the end user to create the application logic and contents, and define interactions without programming. High-level authoring tools [HSGB06] for augmented reality and some toolkits for creating tangible interaction are related EUDs that we review in the next two sections.

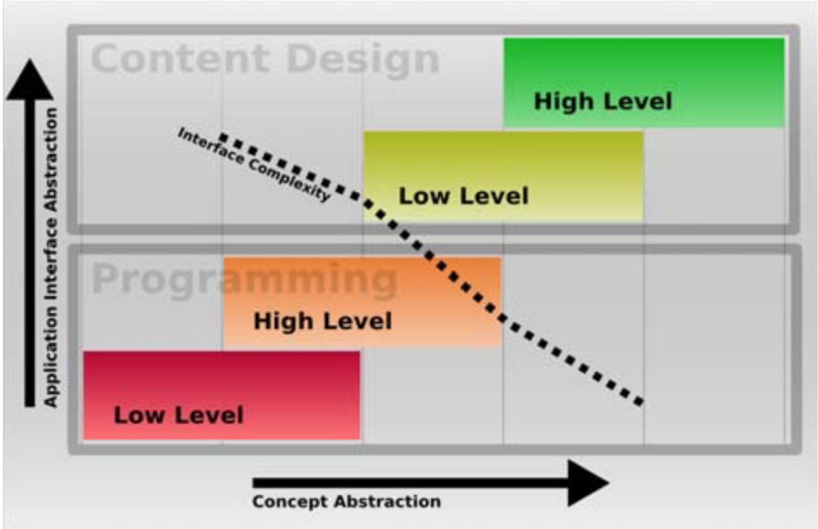


Fig. 2.12 Classification of AR Authoring tools based on the application interface abstractions and concept abstractions [HSGB06].

2.3.1 Authoring for Augmented Reality

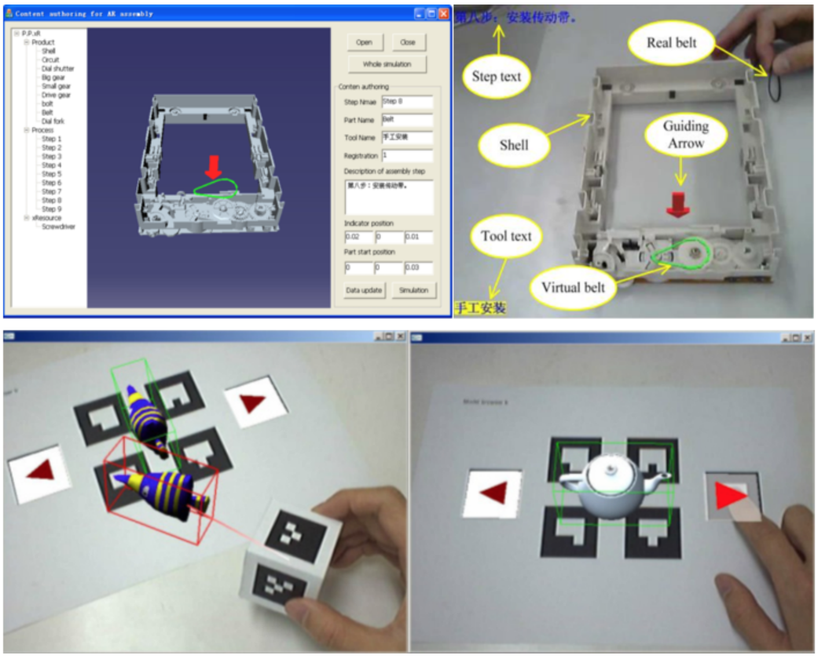
Several research projects have addressed the problem of facilitating the creation of augmented reality solutions. These system are generally known as authoring systems. AR Authoring is in essence involved with creation and managing the relationships between real and virtual entities [Mac02]. Earlier AR authoring systems, such as Studierstube [SFH^{+}02] or DWARF [BBK^{+}01], were actually software frameworks that would facilitate programming of new AR applications. For example both projects offers functionality and abstractions for tracking and multi-user distributed presentations of registered 3D objects on the physical environment. Several applications are build on top of these frameworks, for example SHEEP [MSW^{+}03] and ARCHIE [Mac05] using DWARF and several educational and visualization applications using Studierstube [SSFG98]. However frameworks like Studierstube and DWARF target programmers and using them involves intensive knowledge of software engineering, programming, and augmented reality fundamentals.

Based on a review of previous work, Hampshire et al. [HSGB06] proposed a classification of AR authoring systems based on the concept abstraction and application interface abstraction. Systems at each higher level of abstraction build upon abstractions from lower levels. As the abstraction level increases, the authoring system hides the technical and conceptual details of creating an AR solution from the user. Accordingly, AR authoring systems can be categorized into two major groups: programming frameworks and content design tools (figure 2.12), and in each category we deal with low-level and high-level systems.

Low-level programming frameworks allow creation an AR solution at the lowest level of abstraction and require implementation of core functionalities such as computer vision and computer graphics algorithms. Target user of such frameworks must not only be proficient in programming, but also have extensive knowledge of the underlying functionality such as tracking, interaction, visualization, and rendering. OpenCV [BK00], ARToolkit [KB99], and Open Scene Graph [WQ10] are examples of low-level programming frameworks used to create AR solutions from scratch. High-level programming frameworks build upon low-level programming frameworks, and provide higher level abstractions for different functionalities of AR systems such as hardware abstractions, tracking, interactions, registration, and rendering, all offered in a self-contained framework. The target user of the these systems still needs to be familiar with programming, but the framework hides the majority of low-level technical details of an AR solution and offers them to user in form of higher level abstractions and generalized APIs. Examples of such frameworks are Studierstube [SFH+02] and DWARF [BBK+01].

In content design tools the content and not the programming of AR solution is the focus. These tools offer higher abstraction level in the interfaces and AR concepts, and usually offer graphical user interfaces for configuring the AR solution. In low-level content design frameworks, the description of the application might still be based on scripting and programmatic, however the focus is on the content and configuration instead of underlying technical implementation aspects of AR. For example in APRIL [LS05] an XML-based language is used for defining the scenes, hardware abstractions, tracking, and the content, and UML state diagrams are used for defining transitions between the scenes. Similarly, AVANTGUARDE [San05] uses a data flow approach for description of user's interactions, tracking, and context information and a UI Management System [Ols92] approach based on Petri nets [Pet77] for the dialog control.

Although low-level content design frameworks focus on content and offer higher level abstraction compared to programming frameworks, but still they are too complex in terms of concepts and tools to be used by a user not familiar with computer science and programming. High-level content design frameworks are mostly stand-alone solutions targeted at domain experts that want to create AR solutions without need for programming and without deep knowledge about technical details of augmented reality. High-level content design frameworks offer a graphical user interface and high level, sometimes domain related, conceptual abstractions for creating an AR solution. ISAR belongs to this category of AR Authoring solutions and we focus our analysis of related work in this section on these tools.



Content authoring for AR assembly

Step Nmae: Step 8
Part Name: Belt
Tool Name: 手工安装
Registration: 1
Description of assembly step: 第八步：安装传动带。
Indicator position: 0.02 0 0.01
Part start position: 0 0 0.03

Fig. 2.13 Examples of two different approaches for interaction with high-level AR authoring system: separate authoring and runtime environments [WZLL13] (top) vs. immersive authoring [LNBK04] (bottom).

Beside the classification of Hampshire et al. [HSGB06], there is another classification worth mentioning. Billinghurst [BCL^{+}15] classifies AR development tools into four categories based on the required programming skills: low-level software libraries and frameworks, e.g. ARToolkit [KB99], which require strong programming skills; AR rapid prototyping frameworks, e.g. DWARF and AVANTGUARDE [SK05], which require some programming skills and facilitate creation of AR solutions; plugin approaches, e.g. DART [MGB^{+}03] and Unity plugins, which require skills in the hosting development or design application; and finally standalone AR authoring applications, e.g. AMIRE [DGHP03] or ARIES [WC13], which do not require any programming skills.

There are generally two approaches for interaction with the high-level AR authoring solutions. Most high-level AR authoring systems follow a general architecture consisting of separate authoring and runtime environments (figure 2.13 top). In these systems, the author first designs the application content, interactions between application objects, and user interactions in the separate authoring environment. Then the created AR application is executed in the runtime environment and the end user interact with it. From point of view of the interaction of the author with the authoring environment, this approach can also be called “indirect interaction” [BW19].

The other approach is to use the same AR environment that the end user will be using for designing and creating the AR application (figure 2.13 bottom). This approach is called *immersive authoring* [LKB05] or “in-situ authoring” [SH16]. Immersive authoring is usually combined with tangible AR interaction [KBP$^+$00] and has the advantage of direct 3D interaction with AR application as it is being created. This leads to a faster and simpler authoring process. Lee et al. [LKB05] call this “What You eXperience is What You Get” (WYXIWYG). Immersive authoring solutions are however usually limited in the scope of applications and interactions that can be designed, because they are bound to interaction possibilities in the AR application. On the other hand, indirect GUI-based authoring applications offer more flexibility and versatility in terms of AR application design. Most indirect authoring solutions also provide preview functionality that allows authors to test the AR application as they are designing it, prior to deploying it in the runtime environment.

With the above introduction about authoring systems for augmented reality, we provide a review of selected works in high-level AR authoring solutions. We focus on solutions that require no programming. Some of these solutions however offer a layered authoring approach [SLB08], in which the simpler high-level tasks can be done from the GUI and scripting can be used to describe more complex behavior of virtual objects. We also focus only on systems that use visual tracking (marker-based or natural feature tracking). For each system, we indicated the authoring platform (AP), the runtime platform (RP), and the tracking type (T). For the authoring platform we also indicate if tangible interaction is used for authoring, or an immersive authoring approach is used, where there is no separation between authoring and runtime environments. When there is no clear indication of these aspects in the publication, we use “unclear”. For a more comprehensive listing of high-level AR authoring systems the reader is referred to a systematic survey by Apaza et al. [ATH$^+$18], and works referenced in [San15], [Muñ17] and [RLMT16].

PowerSpace [HR02] is motivated by utilizing the familiarity of technical document authors with tools like Microsoft PowerPoint. The system combines PowerPoint presentation with 3D AR content. The exported AR presentation can be further edited in an editor to integrate registered content on 3D models. The modified presentation is then exported and viewed in a separate viewer (AP: desktop; RP: desktop, VST HMD; T: marker).

Similar motivation is also considered in Designer Augmented Reality Toolkit (DART) [MGB$^+$03]. DART is one of the first and famous examples of plugin AR authoring tools. It was developed as a plugin for the popular designer software Macromedia Director. Target users were designers who are familiar with Macromedia Director and DART allowed them to rapidly create AR solutions. It uses marker-based tracking using ARToolkit and offers AR specific extensions for 3D content, camera control and interaction based on Director's timelines. Designers could use Director's scripting language to fine control the AR scenes and interactions (AP: desktop; RP: desktop, VST HMD; T: marker).

AMIRE [DGHP03, ADG04] provides a component-based approach for creating mixed-reality applications. Generic components can be selected for different functionalities required by the MR application, including tracking, registration, and 2D and 3D interaction and content. The tracking is based on ARToolKit [KB99]. The scenes and their transitions are described using an XML-based language. For each new application the components are configured using their properties and connected together using a dataflow approach. A desktop authoring application is used to configure the components and the communication between them. The authoring environment features a realtime preview of the created scene, that shows how the end user would experience the application (AP: desktop; RP: unclear; T: marker). CATOMIR [HSZ05, ZH04] is a light-weight framework based on AMIRE for authoring MR application with a tablet PC or mobile phone. It offers a light-weight set of AMIRE components and a drag and drop mechanism for adding them to the scene. It also offers dialogs for configuring the component and a visual programming interface for connecting them to define application behavior. Placement of virtual objects and calibration can also be done using tangible manipulation of markers (AP: handheld; RP: handheld; T: marker). Mixed Reality Assembly Instructor [ZHBH03] also uses AMIRE framework. On top of AMIRE's components for object tracking and 3D content, it offers data structures and UI elements for creating step-by-step assembly instructions. The author uses an Authoring Wizard to create the scenes for each steps of the assembly using an "authoring by performance" approach. A tracked mouse allows tangible interaction for placement of virtual objects in scenes. The resulting application is persisted in XML format of AMIRE applications, and executed in Mixed Reality Assembly Instructor application (AP: handheld, tangible; RP: handheld; T: marker).

ComposAR [SLB08] offers a desktop authoring environment for developing and testing the AR scenes and an AR view for mobile phones with Windows Mobile operating system. The author can design the scene by associating virtual objects to markers and configuring them from authoring UI. The behaviors are defined using an event-action mechanism. User's interaction with the markers trigger events which call configured actions attached to scene graph nodes. The authoring environment offers a preview window and possibility to interactively reconfigure interactions at runtime using Python scripts (AP: desktop; RP: desktop, handheld; T: marker).

k-MART [CKL$^+$10] takes a context-oriented approach to AR authoring. The content of an AR application are described as context-behavior pairs. Context is anything related to the physical world, such as presence, location, and orientation of a physical object (marker). Behaviors define augmentations, showing a 3D model or time-based appearance of a text. The AR content are defined using the desktop authoring, including a preview functionality. The application is exported in X3D format and executed in a separate AR browser application (AP: desktop; RP: unclear; T: marker).

Template Based Authoring [KWCS05] is based on the observation that many operations in service and maintenance domain consist of repeating instances of the similar activities, such as opening a screw or placing a part. The systems offers templates for these activity instances. Templates can be configured and added to a 3D scene in a desktop authoring environment. The author configures parameters of the templates, such as location and rotation of a 3D model and arranges their playback order in a timeline editor. The application is exported to an XML format and execute in a separate viewer (AP: desktop; RP: unclear; T: marker).

SUGAR [GTOF12, GMOF13] provides a simple authoring environment where user can place 3D models and multimedia content on photos as scene model. The desktop authoring system allows creation of step-by-step instructions as sequence of scenes. The tracking is done using ARToolkit markers. Scene model can also be taken using a depth camera. In this case the depth information is used for creating occlusion effects. AR application is exported to a ZIP file containing scene transitions, 3D graphics, and other multi-media content, and viewed on handheld AR viewer (AP: desktop; RP: handheld; T: marker).

iaTAR [LNBK04] offers a tangible immersive approach to authoring. The authoring environment is the same as the AR application, and the user defines scenes and interactions by manipulating tangible objects (markers). The application offers three component types: physical objects, virtual objects, and logical boxes. Each component type has set of properties and input and outputs that read or write those properties. Logical boxes are contain pre-programmed application logic, such as arithmetic operators, or complex behaviors like detecting proximity or performing geometrical transformations. Authoring is performed by tangible and logical manipulation of components, such as creating, removing, changing properties, and linking properties together (AP: desktop, VST HMD, immersive, tangible; RP: desktop, VST HMD; T: marker).

ARIES [WC13] takes a model-driven approach based on object-oriented modelling of interactive AR environments [Woj12]. An AR application is defined by configuring, AR-objects, instances of so called AR-classes in a desktop editor. AR-classes are the same as the concept of a class in OO programing, but extended with AR related features such as geometry, behavior, and attached media content and 3D models. Two roles are defined for the authoring, a designer who is familiar with simple programming and modelling using XML, and the domain expert. The designer defines the AR-classes required for specific AR scenario. These AR-classes are then used and configured by domain expert in a simple GUI to create the AR application. The AR application is executed in a separate AR browser (AP: desktop; RP: desktop; T: marker).

Augmented Reality Scratch [RM09] is an AR authoring environment targeted at children. Similar to extension idea of PowerSpace and DART, this tool extends the Scratch [RMMH$^+$09] - a familiar programming environment for children. Having simplicity as design goal, the Scratch environment is extended with AR related functionality in a minimal conservative way. Sprites can be attached and detached to physical objects equipped with markers or interact with simple color-based marker less objects. AR related programming blocks are added to Scratch, that allow child control the behavior of sprites attached to physical objects based on their location, orientation, and relative distance and orientation of markers (AP: desktop; RP: desktop; T: marker).

Wang et al. [WTS10] created an authoring system targeted for educational use that features yes/no questions. The systems also offers rules for transition between scenes based on the answers to the yes/no questions. A desktop authoring GUI allows teachers to create the scenes, including 3D objects, and questions. The AR application is saved in a plain text file, that can be edited by the author, and is viewed in a separate viewer application. The application also features occlusion based detection of touch interaction with the markers representing yes/no answers (AP: desktop; RP: unclear; T: marker).

Simeone et al. [SI11] developed an authoring system based on a content management system (CMS) and marker-less object recognition. The content creators can upload multimedia content to the CMS and associate them with different parts of target image models. In the runtime, object recognition based SURF features and machine learning is performed, and content attached to different parts of the target model are retrieved. Authoring in this system is referred to uploading and editing content in the CMS and not registration of 3D content or creating new object recognition models (AP: desktop; RP: unclear; T: natural features).

ARTalet [HWL$^+$10] offers a tangible immersive authoring environment for creation of multimodal content for magic books (an augmented physical book). The system comprises a tracked manipulation prop, equipped with a vibration motor, for placement and manipulation of 3D models and selection of different functionalities from a 3D menu. Using the authoring environment the users can perform 3D object manipulation (e.g., positioning and rotating, coloring, scaling, coping, deleting), 3D trajectory manipulation, 3D object mesh deformation, and design audio and vibration tactile feedback. A desktop viewer is used to view the magic book after authoring is finished (AP: desktop; RP: desktop; T: marker).

Part et al. [PW09] proposed a multi-layer approach for authoring of magic book. Their system is based on natural feature recognition and tracking of different areas of physical book pages. The author uses the desktop authoring environment to select different areas of the image of the book page using a polygon selection tool. He then assigns to each selected area a layer. The author can assign different multimedia and 3D content to each layer. This way it is possible to create interesting occlusion and transparency effects with the virtual content. The AR application is exported in XML format and viewed in separate AR viewer application (AP: desktop; RP: desktop; T: natural features).

Shim et al. [SKY+14] developed an authoring system for marker-based and gesture-based interaction with 3D objects. The system consists of a simple desktop GUI for associating 3D objects with markers and enabling different hand gestures for them. A live preview shows the 3D objects on the markers. A depth camera is used for hand gesture detection. The user can interact with AR content by manipulating the physical markers or move, rotate, and change scale of the objects using hand gestures (AP: desktop; RP: desktop; T: marker).

Tiles [PTB+01] offers an immersive tangible authoring interface for collaborative tasks. The system consists of markers printed on cards, called Tiles. There are three types of tiles: data tiles can dynamically receive virtual objects, operator tiles allow manipulation of virtual objects; and menu tiles are a catalog repository of existing virtual objects. The virtual objects from menu tiles are added to the data tiles using operator tiles. Operator tiles also allow removing or copying virtual objects, and presenting context and help information. The tiles can be physical placed on magnetic whiteboard. Video see-through HMDs are used as display (AP: VST HMD, tangible, immersive; RP: VST HMD; T: marker).

Langlotz et al. [LMZ+12] created a mobile in situ authoring system for novice users. The system features creation and uploading of natural feature tracking targets to a tracking database. The 3D content can be registered in situ on these tracking targets. Beside text, image, and multimedia annotations, the system also allows creating, modifying, and texturing of simple 3D models using gestural and spatial interaction with the mobile phone (AP: handheld, immersive; RP: handheld; T: natural features).

Rahman et al. [RCES09] created an authoring system for educational purposes. The teacher can select physical objects in a video stream in the authoring environment using a polygon selection tool. A depth camera is used for detecting the user's gesture interaction such as picking or point to an object. The teacher can create and attach different multi-media annotations to the selected physical objects, and upload them to a database. A separate viewer application presents the annotations when the physical objects are detected. It is unclear though how tracking, registration, and object detection is done in this work (AP: desktop; RP: desktop; T: unclear).

2.3.2 Authoring for Tangible Tabletops

Similar to authoring environments for AR, authoring solutions for tangible tabletops can also be grouped based on their target users and the required level of technical expertise in computer science topics. All tangible frameworks, from the early works by Fitzmaurice, Ishii, and Buxton [FIB95], have offered constructs with different degree of abstractions to facilitate realization of tangible interactive applications. We can generally categorize these systems into solutions targeting programmers, and the solutions for the end users without programming skills. In this section we review some of the works in both categories.

Toolkits and Programming Frameworks

Programming frameworks or toolkits provide abstractions on top of programming languages and hardware technologies that facilitate development of tangible tabletop applications for programmers. They are intended for programmer, involve programming and cannot be used by domain experts who are not familiar with programming. On the other hand, they are mostly general purpose, offer a high flexibility in terms of possible design space of the tangible application, and can be used to develop tangible tabletop applications for different domains. Relying on toolkit abstractions, the programmers do not need to have an in-depth knowledge of underlying technologies and algorithms, and can jump-start the development and focus on creating the high-level application logic. These frameworks often include components for hardware abstractions, calibration and coordinate transformations, object detection and tracking, touch detection and gesture recognition, network services, and rendering the content.

Echtler [Ech09] describes a general layered architecture for tangible and multitouch-based interactive tabletops. This architecture devises different functionalities of the system into layers of increasing abstraction. The bottom layer (hardware-abstraction layer) abstracts out the technical specificities of the hardware and provides data about user's actions, such as position, orientation, shapes, and so on. On top of this layer, the transformation layer converts the raw data about user's actions into coordinate space of the display, including the calibration of the system parts. The event interpretation layer, provides high-level application related events by interpreting the transformed spatial data, including detection of touch gestures, an presence and updating of tangibles. Finally, the widget layer provides reusable components for the user visible output. Although not necessarily in a layered architecture, these components can be observed in the toolkits reviewed below at different levels of abstraction.

The reacTIVision framework [KB07], together with its companion TUIO protocol [Kal09], is one the widely used frameworks for development of interactive tangible tabletop applications. The framework is developed as distributed application, with the a tracking library at its core. The tracking library is optimized for fast and robust tracking of spatially designed fiducial markers, touch interaction, and bounding box of blob shapes. The state of the table, extracted from the tracking information is continuously transmitted using TUIO protocol over UDP packets to TUIO clients. A TUIO client then turns the state messages into for tangible application more useful higher abstractions and events such as object added, object updated, or object removed, as well as touch interaction. The application logic of a tangible application is then implemented in the callback function to these events. The framework also includes low-level functionality needed for camera-projector calibration and correcting image distortion. This loosely coupled design, allowed the development of TUIO clients in different programming languages, that has lead to wide spread of applications based on reacTIVision. Several toolkits for tangible development are also based on reacTIVision as will be described below.

ARBlocks [RdFST13] is a framework for creation of projective augmented reality and tangible applications based on physical blocks. Target domain of the framework is early childhood educational applications. The system features physical blocks of about 6x6 cm size. Each block has a specially designed marker pattern around its edges that bounds an empty area, on which information is projected. The framework consists of tracking library for tracking these blocks, the calibration module for camera-projector calibration and the projection module for registering the content on the blocks. On top of these three basic modules different applications can be developed. The users (children) can interact with the application by manipulating the blocks. For example in an application for learning vocabulary, images of the animals are projected on one set and their names on another set of blocks. When the child puts the block with the corresponding name near the correct image, the system gives feedback by highlighting both blocks with green.

TULIP [TML15] is a framework for rapid prototyping of tangible tabletop applications. The authors argue about the lack of extensible frameworks for TUI development that allow reuse of abstraction in the way offered by GUI widget frameworks. Based on this argument, they propose a framework for rapid application development of tangible tabletops applications that offers modularity and a reusable abstraction of TUI widget as the core concept. Based on the original model of MCRit (Model-Controller-Representation [tangible/intangible]) by Ulmer and Ishii, a widget in TULIP framework has a tangible physical part called handle and an intangible part called corona. There exists a hierarchy of different widgets that all inherit from the base widget abstraction that takes care of physical context of the widget, such as position and rotation, and its connection with the application logic. There also exist a hierarchy of different coronas, such as text box, pointer, image, infobox, and shadow that inherit from base corona class. The system relies on reacTIVision and TUIO for compute vision and tracking tasks. It is tested with different applications build for a back-projected tabletop.

PapARt [LH12] is a camera-projector hardware and software system for immersive drawing and manipulation for projections on a physical paper. The hardware consists of a short-throw projector, an RGB camera, and a depth camera packages in column-shaped frame that is put on a table. The system uses marker-based tracking for tracking the physical paper as drawing surface and uses a depth camera for detection of touch and 3D over surfaces pointing. The user can create virtual drawings on the physical paper using finger movements, or can physically draw based on projected guidance. Another tracked piece of paper with four markers is used as the menu to call different functionality of the system. The functionality of the system are demonstrated using different applications scenarios: projections are used to guide users in physically drawing a 2D image; users can use gestures to pan and zoom information projected the drawing area; users can move the drawing area to change light intensity of the projection or use point gestures to place a virtual lighting; rotating the drawing area can change the 2D projection of a rendered 3D model according to user's perspective. The software offers APIs for Processing and plugins for Unity.

TACTIC [NRD15] provides a browser-based API for creating tabletop applications that feature touch, tangible, and over surfaces interaction. The system relies on reacTIVision and TUIO for touch detection and marker-based object tracking, and on library for 3D gesture recognition based on depth data from a Kinect camera. The system is developed as a node.js application and uses RabittMQ for communicating 3D gestures and binding other components, that don't use TUIO protocol. In order to use the API, a developer needs to add corresponding CSS classes to HTML elements. For example the class "touchable" adds touch behavior to an HTML element. HTML events are also used for communicating between the API and tangible tabletop application logic. The developer of the application binds asynchronous JavaScript callback functions to HTML events in the familiar way done in web development. For example binding a function to the “object.added” event of the surfaces application, the developer can perform his desired application logic, when an object is added to the surface. The event callbacks receive an argument that contains event data. For example in the case of object.added event the id and 2D coordinates of where the object was placed. A similar approach is also taken by Ubi Displays toolkit [HA12] that expose multi-touch events to content created as HTML elements and application logic implemented in JavaScript in a web page.

ToyVision [MCB12] is a software toolkit for rapid prototyping of vision-based tangible tabletop games. The system is built on top of reacTIVision framework and TUIO protocol [Kal09] and adds high-level abstractions by offering a widget layer. The widget layer is a layer on top of the the “event interpretation layer” [Ech09] and is specifically designed to detect the token commonly used in games. Four types of tokens are distinguished: simple tokens, named tokens, constraint tokens, and deformable tokens. ToyVision extends marker-based object detection of reacTIVision for identifying these types of tokens. The widget layer receives raw-events from TUIO sockets and creates high-level abstractions that can be directly used to define game logic without the need to interpret raw-event data. The widget layer and the high-level events and abstractions related to tokens significantly reduce the amount of code needed to create a tangible game compared to using raw events of TUIO and reACTivison.A GUI is used to add and configure tokens to be used in the game. For each token type the designer configures information needed for detecting and using the token in the game logic, such as the size of the blob and the name of the token. The editor automatically generates XML description of the tokens that are used to generate high-level events. Simple, named, and constraint tokens are tracked using fiducials. The deformable tokens are detected based on the minimum and maximum size of their blob.

Papier Mache [KLLL04] is one of the early frameworks to address the technical challenge of developing TUI applications by providing high level event-based abstractions independent of the input technology. Using Papier Mache developers can do rapid prototyping of tangible applications that utilize RFID, vision, or barcode input. The system created events related to presence, removal, and modifications to the state of tangible objects in the scene. The developers can associate these events with corresponding actions such as playback of an audio clip, fast forwarding or rewinding a video, or opening GUI windows. Papier Mache simplifies the process of development of TUIs by providing these high-level abstractions that allow creation of simple applications in some few lines of code. It also feature a debugging GUI and mock event generator that help in testing and debugging the application. It is however not specifically designed to address visual output using projection on a tabletop, and is more focused on associating tangible inputs to application logic, for example a home automation system.

End user Development Environments

EUD environments provide GUIs that are used by domain experts to create applications based on their needs. These environments offer domain-specific concepts and UI metaphors and often are designed to take advantage of the familiarity of the authors with existing digital tools and workflows. They are mostly targeted for a specific domain (e.g. game development, education, rehabilitation) and offer functionality tailored to the requirements of the domain experts. Accordingly, they are less flexible and versatile in terms of possibilities of designing the application and provide mostly fixed workflows for designing application content and limited configuration possibilities for designing the interactions for the final end users. Please not that the term “end user” can be used in an exchangeable meaning depending on the context: sometimes as the domain-expert who uses the EUD to create the tangible application, such as a teacher or a physiotherapist; and sometimes as the final users of that application, such as students, or physiotherapy patients. Tetteroo et al. [TSM13] discuss different challenges of creating end user development environments for tangible interaction. These challenges include integration of physical and virtual worlds, preparation of tangible artifacts, support for design of interaction by the application authors, support for end users without technical affinity, and considering the soci-technial context of the tangible application. In the following we review some of the EUD environments for creation of tangible tabletop applications by non-technical end users.

TagTrainer [TVG$^+$15] is an authoring environment for physiotherapist to create arm-hand rehabilitation exercises based on tangible interaction (figure 2.14). The system uses special interactive boards called TagTile boards. TagTile boards are a grid of backlit LED cells that can detect the presence of an object and track its position on the grid. The detection and tracking are based on RFID. Using the authoring environment the therapist can define rehabilitation exercises as a sequence of tangible interactions with the tracked objects on the board. For that, the therapist defines a timeline consisting of actions including lift, place, and move objects. The timeline can include multiple objects simultaneously and also includes actions for providing instruction and pausing. For each object related action, the properties of the action can be defined, e.g. the position on the board to place the object.

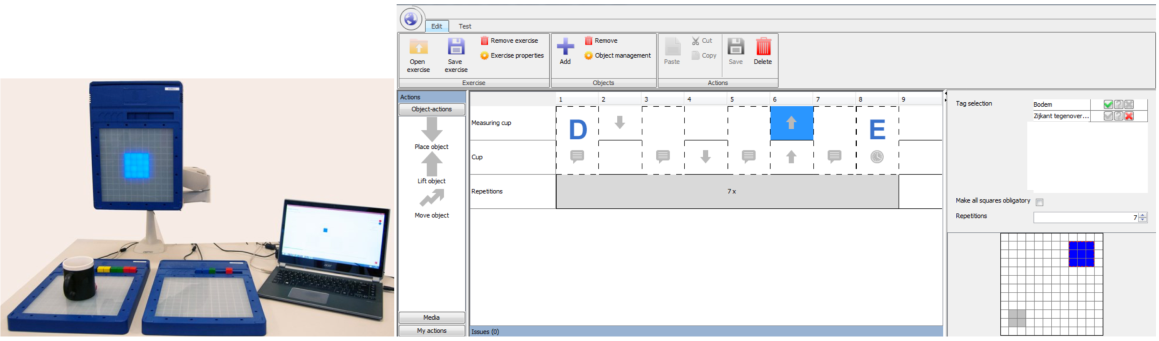


Fig. 2.14 TagTile boards and the Tag Trainer authoring environment [TVG+15]. The physiotherapist defines an application as a sequence of object placement and moving and audio-visual feedback. Objects are detected and tracked by RFID.

COPSE [MTA+17] is a system developed for lowering entrance barrier for educational experts for creating, modifying, and reusing tangible tabletop applications for microworld [Edw95] scenarios. Microworlds are artificial environments that behave based on mathematically expressed scientific models. These rules are however hidden from the learner and he must discover them by manipulating objects and interpreting feedback. The microworld in COPSE are described using a series of mathematical equations that relate input (independent) variables to output (dependent) variables. The system is build using TULIP [TML15] frameworkd and offers three types of building blocks: widgets for providing input and localized feedback, equations for defining the model of the microworld, and scenes for visualizing the feedback. The application is described using an XML-based language. The variables in the equations that describe the microworld are represented using tangible widgets of TULIP. There are two types of widgets: on/off variables are represented by placement of the tangible on the tabletop; the continuous variables are represented by rotating the widget. Coronas are attached to widgets and represent the feedback of the system. Their placement is relative to the widget's handle. Text, image, shadows, and gauge, are examples of coronas. A scenario consists of multiple scenes that are bound to output variables. The value of variables determine if the scene is rendered (trigger condition) and its representation. Currently the system does not provide a GUI for creating an application and the XML file describing an application must be created manually.

DEDOS [RÁMH+18] is a system for creating educational card-based content and activities by teachers. The system consists of DEDOS-Editor, and editor for creating the content, and DEDOS-Web, a web-based player for playing back the content on different devices such as tablets, interactive whiteboards, and interactive tabletops. It also includes a learning analytics that logs the interaction data as students are working with the content. The design of the editor is based on the types of the educational activities. Roldan-Alvarez et al. [RÁMGHH16] found out the most demanded activities that teachers need an authoring system to support, are single and multiple choice questions and pair matching activities. Accordingly, the DEDOS-Editor is designed to allows creation of these activities. Additionally an activity for connecting points, and an addition activity are supported and the activities can also be combined together. The type of activity defines the application logic for controlling and presenting the result at runtime. The teacher defines correct answers by adding so called goals to the activity. For example for a multiple choice activity, he adds the correct answer goal to the card corresponding to the correct answer, or for connect the dots activity, he adds goals to the dots representing the correct paths. All activities are based on a UI metaphor of cards. The teacher designs the content of the cards, including text and graphics and adds them to the activity area. For multiple choice questions, the teacher can select which cards are the correct answer. For pair matching activities, the teacher matches the correct pairs using arrows. The teacher can also design the layout of the activity when it is presented to the students by defining main game area, and individual areas for each student. The created application is saved as a ZIP file containing an XML file and the media resources. The application is then played back in the DEDOS-Web player application. Before a project is started, DEDOS-Web allows configuration of the content to support different learning scenarios. For example the teacher can configure groups for collaborative working on the activities, define the navigation flow between activities, or enforce timeouts and feedback for moving to next activity. The player application also features a responsive UI and supports adoption of the content to different device size and resolutions, so that the activities can be played back on different devices.

EMIL [LHLD13] is a rapid prototyping tool for creating UI of interactive tabletop applications. The target users are teams of designers and programmer, that collaboratively create the UI using the immersive authoring environment and a library of UI components. EMIL consists of three main parts: a UI framework with a component library, an authoring tool, and a reusable database of components, applications, and design knowledge created from previous projects of the design team. The system is created using TUIO AS3 framework [LBH$^+$10] that adds gesture interaction to an application built on top of TUIO and reACTivision for marker-based object tracking and touch detection. The UI framework consists of visual components and non-visual components. Visual components include widgets like lists, labels, buttons, images, etc., views, which contain widgets, and the application templates, which consist of views with transitions. Non-visual components are controls, which are gestural inputs such as dragging, rotating, and scaling, and behaviors which are complex functionality that are created by connecting interactions with actions. For example a component can have “trash bin” behavior, which combines dragging over, dragging out, and releasing with shrink, restore, and remove actions. That is, when another component is dragged over the component with trash bin behavior, it’s representation is shrinked, if it is dragged out then its representation is restored, and if it is released then it is removed. The EMIL’s authoring application allows immersive authoring of UI of interactive tabletops. It offers two modes: live mode and authoring mode. In authoring mode the designers can add widgets and view to surface from menus. Each component also shows extra menus that allows configuring its properties. A project consists of resource and configuration files that are stored in a shared cloud storage, to allow parallel working on the design of the UI and creation of resources.

Thevin et al. [TJR$^+$19] developed an end user programming for teachers of special education for creating interactive audio-tactile graphics for visually impaired persons. Using this system the teachers without technical background can easily augment their existing educational tactile media with audio information. It uses an immersive authoring approach, by superimposing digital feedback on tactile media when creating the content. Two main use-cases are content creation by teachers and content exploration by visually impaired persons. Content creation is done by associating audio information to selected areas of the tactile map. The system is based on an spatial augmented reality camera-projector setup and an existing framework for physical AR drawing (PapARt [LH12]). Using PapARt framework and setup, the teacher directly augments the tactile map with interactive areas by touching the corresponding area on the map and recording audio information. To designated the interactive areas, the teachers draw with finger directly on the tactile map. To give visual feedback to the teacher, the drawn shape is projected directly in realtime on the map. Different functions of the application are called by pressing corresponding keys on on the keyboard of the computer running the application. Functions include drawing line and shape, recoding and stopping the microphone, deleting audio annotations or a whole interactive area, and changing to explorations mode. The same setup is also used for playing back the content in content exploration mode. The visually impaired person can explore the tactile map with both hands. Touching an area with one finger triggers playback of the associated audio material.

KitVision [BMBC19] is an EUD environment for tangible tabletops targeted at physiotherapist (figure 2.15). The system supports creation and customization of cognitive therapy exercise by the therapist and is specifically designed for cognitive therapy exercise and offers a GUI to create and modify these exercises. The GUI is based on a model of cognitive therapy exercise expressed in a XML-based language. According to this model, a therapy activity consists of an ordered sequence of tasks. Each task has a background projected on

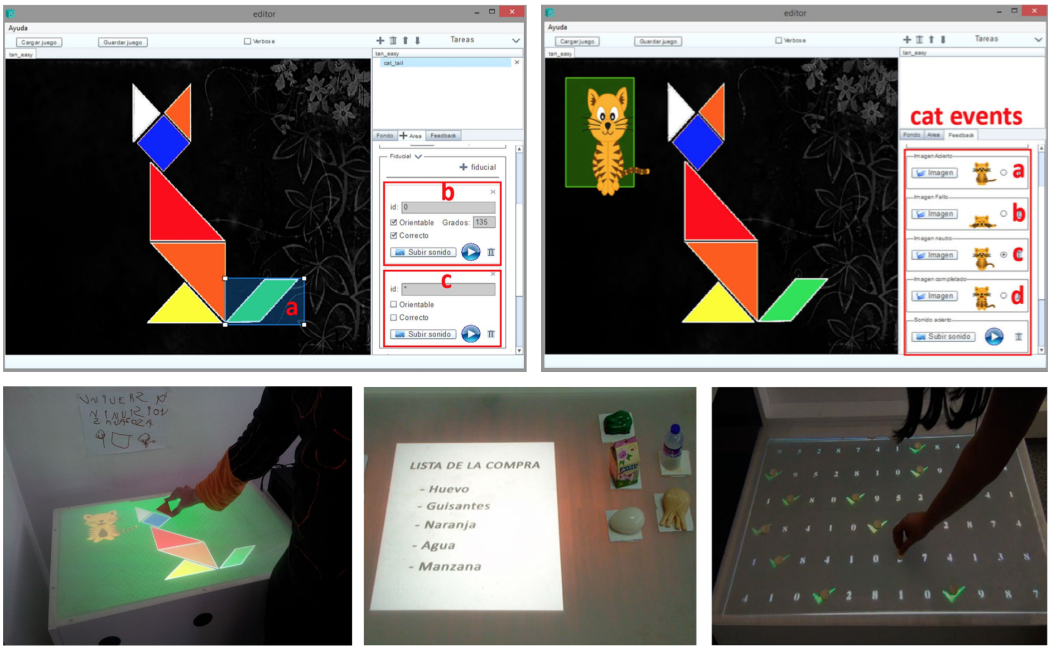


Fig. 2.15 KitVision authoring environment and three example applications (Tangram, shopping list, counting) [BMBC19]. Authoring is done by defining interactive areas and associating a list of objects and feedback content to them.

the tabletop. Associated with each task are a set of "interactive areas" and "playing pieces". Placing a playing pieced in an associated interactive area triggers different events. Outside the area the pieces have no meaning. The areas are rectangular and can be placed at a fixed position on the table or connected to an object. The triggered events are used to define feedback. Each task also has an associated feedback, which is graphic and/or audio element. Each area has a list of wrong and correct physical objects. Physical objects can have sounds and orientation associated with them. The feedback is based on if all the correct objects are placed in an area. The system should detect different objects and also instanced of the same type of objects. The desktop authoring environment is used for creating the application and the KitVision Player is used for running it for the patients. The applications run on the NIKVision [MBC13], a back projected interactive tabletop. The system uses reacTIVision [KB07] framework for marker-based object detection and tracking. The authoring process consists of defining interactive areas for each task of the activity. Then for each a list of correct and wrong objects are defined. An area can also have a correct orientation for the object defined, in which case only one correct object can be assigned to the area. In the third step, the feedback for each are is defined based on three states of the placement of the objects in the area: incomplete waiting, wrong, complete. After defining all tasks in the activity it is exported to the XML format with all the resources to a USB disk. The activity is then opened automatically in the KitVision player by inserting the USB disk in the NIKVision tabletop. The authoring environment does not feature a preview function and the author must run the application in the player on the tabletop for testing it. The feasibility of the cognitive therapy model and the system is shown by developing different exercises: A counting game, a select correct item game, classical Tangram game, a memory game, and a complete the sequence game. However, being based on the XML model of cognitive therapy exercises, the system has a fixed and rather inflexible workflow for defining each scene. For example, at the start of each activity the author must define a background image is displayed full screen, and must define an audio instruction and an icon. Configuring the scene and designing it freely is not possible. Similarly the events and interactions are only limited to correct or wrong placement of the objects, and a it is not possible to flexibly define interactions by connecting different events with different actions. Also, lacking a preview or immersive authoring functionality makes the development process cumbersome: the author must create the application on the desktop authoring environment, export it to USB disk, and then test it on the tabletop, and repeat the process for every modification.

Chapter 3

Requirement Specification

This chapter describes the use cases and requirements of ISAR based on the literature review from previous chapter, our observations and experience from related projects (see 1.2), and the visionary scenarios. We first describe in an as-is scenario from the domain of vocational education, the context in which ISAR could be utilized. We then describe in different visionary scenario how ISAR will be used to address similar problems. In scenario-based requirement elicitation a visionary scenarios describes the functionality that the envisioned systems will provide [BD03] and how it will influence the work practice of the target users. We then formalize the requirements described in the visionary scenarios in use-case model, functional requirements, and non-functional requirements.

3.1 As-Is Scenario: Teaching Mainboard Assembly

John is a teacher for elementary level courses in a vocational school for electronics. In one of his classes, he teaches the students how to assemble, troubleshoot, and repair personal computers (PC). One of John's lessons is about assembling a mainboard.

At the first step of the lesson, the students must learn the names of the different components and the theoretical information about them. So far John used a PowerPoint presentation to teach the different hardware components to the students. His slides show name and positions of different parts of the mainboard, describe different hardware components, such as memory chips, CPU, graphic cards, that should be assembled on the mainboard. The slides also contain hints and important aspects that the students need to pay attention to, for example the correct orientation of CPU when inserting it in the CPU socket on the mainboard.

After the theory parts, the students start assembling the model mainboards in the practice part of the lesson. For each step of the assembling workflow, John has printed a paper instruction sheet that has a text instruction and multiple pictures. The pictures show the hardware component and where and how to assemble it onto the mainboard. They also include the hints and warnings for things that students need to know. During the practical part of the lesson, in order to assess the students, John moves between groups and ask questions. For example, he asks the students questions like, "Show me the CPU" or "Show me where is the GPU socket".

The above approach however has some shortcomings. The theoretical part of the lesson is disconnected from the practical part. Accordingly the students have difficulty applying what they have learned in context of the real task. They don't know the names of the hardware components, forget important hints, and don't know how to properly use tools. They constantly make mistakes or have to ask John for help. Furthermore, in the practical prat, going back and forth between the task context of assembling the mainboard and the paper instructions is distracting for the students. Additionally, if John notices a problem with the paper instructions, he has to edit and reprint all of them. Another problem arises when there are a lot of students in the classroom. John cannot attend every group for assessment and asking questions.

John would like to address the above problems in his class. He is also interested in new educational technology. He has recently read about interactive tabletops and augmented reality, and would like to try it to improve his class. Specifically, he would like to use an interactive tabletop, because it allows for hands-free operation on the table that is needed in his lessons. He wants a system that allows him define learning content as digital content projected on the physical world. He also wants the system to be easy to use and flexible enough to allows him to experiment with new content and adapt the contents he has created after observing how his students interact with the setting.

However, after talking to his friend, who is a programmer, he realizes that he cannot create content for his lecture on his own, and needs a programmer to help him. Furthermore, he realizes that despite the high effort for creating the content, these content can only be used for one of his lectures, and creating new content is associated with a lot of effort again. John's friend tells him, that although there are authoring systems for end users to create content for augmented reality, these systems are limited to mobile or desktop augmented reality.

3.2 Visionary Scenarios

ISAR can be used by the end users and expert in different domains like education, rehabilitation, and manufacturing in order to create interactive augmented reality tabletop applications for their specific needs. We describe a visionary scenario for each of these exemplary domains.

3.2.1 Situated Learning of Taxonomies and Domain-specific Vocabulary

Learning domain-specific vocabulary and taxonomies, i.e. learning names of the objects and concepts, is the essential elementary step in learning any activity. The theory of situated cognition [RA09] describes how gaining knowledge, such as learning vocabularies, can benefit from being embedded in the context of the activity. Referring to the above as-is scenario (3.1), we describe here how ISAR can be used to support situated learning of domain-specific vocabulary.

ISAR helps John, the teacher, to create the learning content for the mainboard assembly task. The first step is to teach the students names of different hardware components such as CPU, RAM modules, GPU, heatsink, thermal paste, etc. John creates a project in ISAR's Authoring System and names it 'Mainboard Components'. For each hardware component, he adds a scene to the projects that contains information about that component. For example for the mainboard itself, he creates a scene with the name 'Mainboard'. He adds the mainboard physical object to the scene. He add multiple arrow annotations that point at different parts of the mainboard and show their names, for example for CPU socket, for RAM sockets, and so on. He adds a video annotation to the scene that shows different models of mainboard and how mainboards are manufactured. He also adds multiple audio annotations to the scene that pronounce the names of different parts. He also adds buttons for moving back and forth between scenes.

After defining the content of the mainboard scene, John defines the interactions of the students with the scene. For example, he can define in interaction rule like 'If student points at CPU socket, playback the audio annotation for its name', or defines a rule like 'If the student selects the video annotation, pause the video'. By combining annotations, physical objects, and interactions, John can also define scenes for exercises. For example, he can add multiple text and multiple checkbox annotations to created a multiple choice exercise. Or he can create a scene containing the text annotations that asks the student to show the CPU socket, and when the student points at the CPU socket, an image for positive feedback is shown and a sound is played.

John continues adding scenes for hardware components and tools involved in the mainboard assembly task. Each scene can contain multiple annotations and multiple physical objects, and many different interaction rules that govern the interaction of the students with the table. He can configure the attributes of annotations and interaction rules for his purpose. The Authoring System of ISAR allows John to define and test the scene exactly as they would appear for the students. This is called WYSIWYG or What You See Is What You Get. John can directly test his scene definition in the Authoring System and change them as needed.

After John is finished with the definition of the scenes, he loads the project into ISAR's Training System for the students. The Training System projects the scenes on the table and reacts to the students interactions according to the defined interaction rules. For example, in the mainboard scene, when the student puts the mainboard on the table, the system recognizes it and projects the arrows that show different parts of the mainboard. The student can use a tracked stick to select mainboard parts. For example, when he selects the CPU socket, the audio annotation for CPU socket is played. Every scene is projected on the table exactly as John had defined and all the interactions with the scene happen based on the rules he has defined and tested.

Adding new Physical Objects

In order to project digital information on the physical objects and in order to detect the interactions with the objects, ISAR needs to recognize the objects. The first step of using ISAR is adding the new physical objects to the dictionary of the objects detected by ISAR. For this, John has two options: marker-based and markerless approach. In the marker-based approach, John attaches marker to each part and tool, and uses the ISAR's UI to add the objects to ISAR's detected objects, by simply selecting the object (with attached marker) and giving it a name. Marker-based approach is a fast method for adding new objects to the objects detected by ISAR. However it has the disadvantage that exactly the same markers must be attached ot the parts and tools that students use for their training. The other approach for detecting the objects is the markerless approach. This approach relies on machine learning models to detect objects. The models for detecting the objects must be trained with a dataset consisting of multiple labeled images of the objects. ISAR offers a UI for training these models. In a video stream John can select the objects give them labels. ISAR then creates a dataset of the selected objects and trains the required models. This process has the advantage of not needing the markers, however the training of the models might take a long time, up to a day.

3.2.2 Guidance through a Workflow

ISAR can also be used to guide a user through a physical activity that is structured in the steps of a workflow. We refer again to the as-is scenario described in 3.1.

A workflow is the sequence of steps needed to achieve the task goal. Each workflow step involves one or more physical objects, e.g. tools and materials. Each workflow step also has an instruction and possible important information that need to be noted. In ISAR each workflow step is associated with a scene. A scene is a combination of physical objects and annotations. The annotations show the instructions and important information that are needed for the step to be done.

John uses the ISAR's Authoring System and defines a scene for each step of the mainboard assembly workflow. He creates a project and names it 'Mainboard Assembly Workflow'. For each scene he adds the corresponding physical objects, e.g. the mainboard and RAM module for 'Inserting the RAM' step, or mainboard and CPU for 'Inserting the CPU' step, or mainboard, heatsink, and screwdriver for the step corresponding to fixing the heatsink on CPU. For each scene he also add the corresponding annotations, such as a textual instruction for the step, highlighting the component that should be picked, highlighting the place where each component should be inserted, a video annotation that shows how to place the component, and so on. He also adds buttons to each scene for navigating through scenes. After defining the scene content, for each scene John also define the interaction rules. For example 'When RAM module is picked, highlight the RAM sockets on the mainboard', or 'When CPU is placed at CPU socket, show the text instruction for locking the CPU and start the video that shows how to lock the CPU'. John can also define similar rules for implicit automatic transition between scenes, or the student can explicitly select the scene navigation buttons added to the scene.

After John is finished with defining the learning content for the workflow, i.e. defining the scenes, interaction rules, and navigation flow between scenes, he loads the project into ISAR's Training System for the students. The training system projects the scenes on the student's workbench. Each scene corresponds to one workflow step. All annotations are shown and the interaction of the student with the table, including the transition between scenes, follows the interaction rules defined by John.

Adapting the Content

After observing how the students learn with the tabletop, John realizes that his learning content for mainboard assembly requires some adaptions and improvements. Some of the instruction text are not detailed enough; Some of the annotations highlighting the parts of mainboard should be in a different color; and some of the interactions, such as scene transitions, confuse the student.

In order to change the learning content, John loads the project into ISAR's Authoring System. For each scene, he can select the annotations that he wants to change and change their attributes. He can also change the interactions, define new ones and remove the ones that confuse the students. He can also change the order of the scenes for the workflow steps and add new scenes for more detailed workflow description.

3.2.3 Rehabilitation for Hand Motor Skills

Hand-eye coordination is an essential part of manual-activities. Patients who survive a sever stroke usually lose their motor skill and hand-eye coordination abilities [HKD+13]. Rehabilitation of these abilities requires a long-time treatment involving different physical exercises, such as grasping, reaching, tilting the wrist, and pointing that are performed under supervision of physiotherapist. The following visionary scenario shows how ISAR can be used by physiotherapist to create motor exercises that a post-stroke patient can perform on site or on his own at home.

Marry is a physiotherapist for post-stroke rehabilitation. Susi is her patient that has recently survived a sever stroke and has lost her motor skills to a large extent. Marry wants to use ISAR to create two kinds of exercises for Susi. The first exercise improves Susi's hand-eye coordination and the second one improves her reaching abilities. For the hand-eye coordination exercise, Marry creates a scene and adds a curve annotation, a timer annotation, and a feedback annotation to it. In the execution mode, this curve is projected on the tabletop and Susi is supposed to follow the path of the curve using a tracked stick, and receive feedback on how she has improved her motor skills. The feedback is calculated based on the time it takes for Susi to follow the path, and the deviation of the tracked stick from the path. For the reaching exercise, Marry adds multiple animation annotations to the scene and

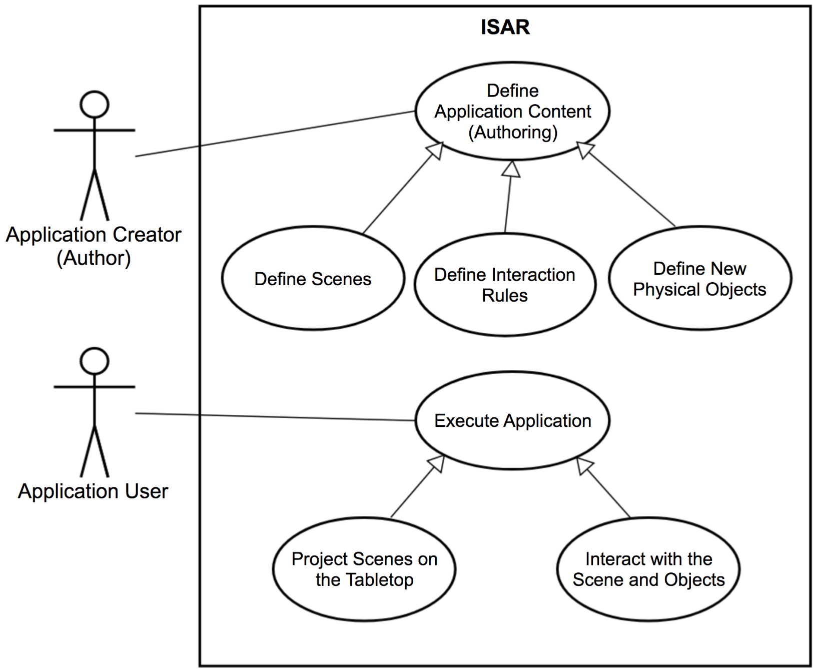

ISAR
Define Application Content (Authoring)
Define Scenes
Define Interaction Rules
Define New Physical Objects
Execute Application
Project Scenes on the Tabletop
Interact with the Scene and Objects
Application Creator (Author)
Application User

Fig. 3.1 High level use cases of ISAR. The application creator (author) defines application content and interactions. In the execution mode the application user sees the content projected on the tabletops and interacts with the application and physical objects.

configures their path and speed. An animation annotation shows an image that moves along a defined path with a defined speed. In the execution mode, the animations are projected on the tabletop and Susi is supposed to hit the image with her hand. Based on how many images Susi hits in the given time, she receives a feedback on her progress.

3.3 Functional Requirements

ISAR has three actors: the application creator (also called author), the application user, and the framework extender. ISAR also has two modes corresponding to the two actors application creator and application user (figure 3.1). In the *Authoring Mode*, the application creator uses ISAR authoring environment to create interactive augmented reality applications for the tabletop based on his specific requirements. In the *Execution Mode* the content of the application is projected on the tabletop and the application user interacts with the application and objects on the table.

3.3.1 Creating ISAR Applications

The application creator (author) uses ISAR's authoring tool to define the application content (figure 3.1). An ISAR application is a combination of physical objects, digital content that are projected on the tabletop, and a set of interaction rules defined by the author. The application user interacts with the physical objects and digital content according to the interactions rules, that govern response of the system based on the actions of the application user.

In order to create an ISAR application, the application creator must define scenes, define interactions, and define physical objects:

Define Scene

The application content are projected on the table as scenes. Figure 3.2 shows the use cases for defining scenes and workflows. A scene is combination of digital content, the annotations, and physical objects. A rich set of different kinds of annotations should support the application creator in designing the scene. These include simple geometric shapes like line, rectangle, circle, curve, and arrow; multimedia content like text, video, audio, and image; dynamic annotations like timer, counter, and animation; and interaction elements like touch buttons, checkboxes, object placement areas, and feedback annotation. The application creator designs the scene by adding physical objects and annotations to it, and configuring attributes of annotations. The annotations can be fixed on the scene, or attached to physical objects. In the later case, the projection of the annotation on the tabletop should be adapted to the position and orientation of the physical object.

Workflows can be defined by adding multiple scenes and defining the ordering and navigation rules between them. Each workflow step is associated with a scene and the transition between steps happens explicitly or implicitly according to interactions of the application user with the scene. In order to add new physical objects to ISAR, the application creator can train new object detection packages and include them into ISAR, or create new marker-based object detection and tracking packages. In this way, the tabletop can recognize new physical objects.

Define Interaction

The table should respond to the events that result from interactions of the application user. Several interactions are possible on the table, for example picking an manipulating objects,

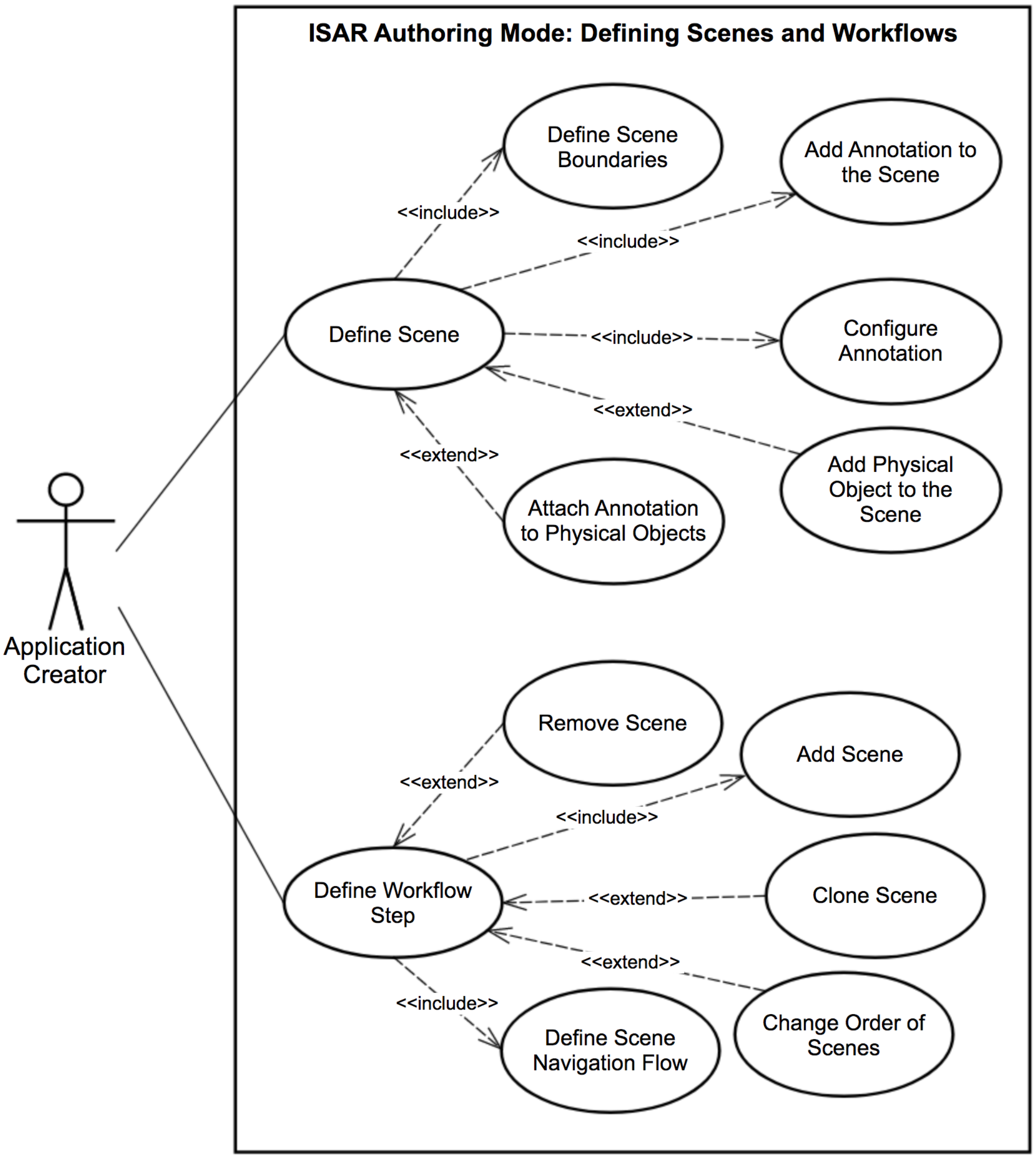


Fig. 3.2 Defining Scenes and Workflows in ISAR authoring mode. The application creator defines the scene by adding annotations and physical objects. Workflows are defined as a sequence of scenes and the navigation rules between them.

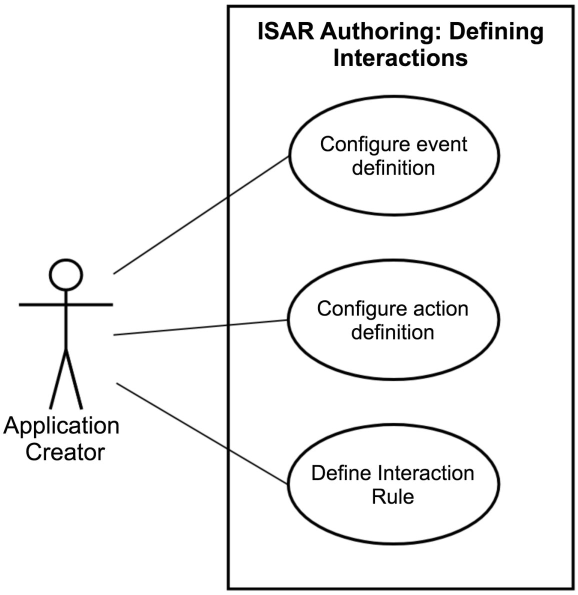

ISAR Authoring: Defining Interactions
Configure event definition
Configure action definition
Define Interaction Rule
Application Creator

Fig. 3.3 Interaction definition use cases in ISAR authoring. The application creator can define interactions by configuring events and actions and defining interaction rules that connect events to actions.

selecting annotations and objects, or timeout of a timer. An interaction can be defined as an event-action rule, that triggers an action when the event is occurred. The application creator can configure the events and actions and define these interaction rules (figure 3.3). For example, the application creator can configure rules like "if the correct checkbox is selected, show positive feedback" or "if the timer is timed out and the correct object is not selected, highlight the correct object", or "highlight a specific physical object as soon as the workflow step is entered".

By combining events and actions as interaction rules, the application creator can exactly define how the table responses when application user interacts with it. The interaction rules also allow the application creator to define different kind of exercises like multiple choice questions in which the application user should select one or multiple checkbox options, object selection exercise in which the application user should select one of the physical objects present on the table,, or object placement exercises, in which the application user should put a physical object in a specific area on the tabletop.

Define new Physical Objects

An ISAR application consists of scenes that are combinations of physical objects and virtual content. Before physical objects can be added to a scene, the tabletop needs to recognize and track them. ISAR supports both markerless and marker-based object recognition and tracking. For any of this modes, prior to defining the scenes, a corresponding object recognition and tracking package must have been added to the ISAR authoring application, that offers recognition and tracking of the required physical objects. A physical object has a name a template reference image that is used in order to define the scene. The registration of annotation on the physical objects in the execution mode is based on object's orientation with reference to this template image (model-based tracking). For marker-based object recognition and tracking physical objects must be associated with markers. For markerless object recognition and tracking a machine learning model must be trained (figure 3.4).

For marker-based object recognition and tracking, the application creator can use ISAR's authoring GUI to associate physical objects to the markers. The advantage of the marker-based approach is that it is simple and fast, and the application creator can add physical object definitions to ISAR immediately before defining scenes. The disadvantage of this method is that exactly the same marker must be attached to the physical objects in the execution mode.

For markerless object recognition and tracking a machine learning model must be trained based on a training dataset of physical objects. This approach is more complex and less robust than the marker-based approach, however it has the advantage that no markers must be attached to the objects in the execution mode. For training object recognition models for new physical objects the application creator can user ISAR authoring GUI in order to create a labeled training set for physical objects. The application creator puts the objects on the tabletops and selects them in a video stream of the tabletop. The application creator only needs to select a few sample images and assign labels to them. ISAR then creates a larger training dataset by combining different training images and applying data augmentation methods. The created data set is then used to train a machine learning model for object recognition. The training however can take a longer time, and as soon as the model is trained with desired performance, the application creator receives a notification and can download the trained model and integrate it into ISAR. After the new object recognition and tracking package is integrated into ISAR, the physical objects detected by this package are available for defining scenes. Relying on natural features, a robust markerless object recognition and tracking puts some constrained on the properties of the objects that can be detected as well as on the lighting conditions. The objects should not be too small (roughly less that 5 cm) and the different object classes must not be too similar to each other and have distinctive features. The lighting condition in the execution mode must be similar to the lighting condition as the training set was created.

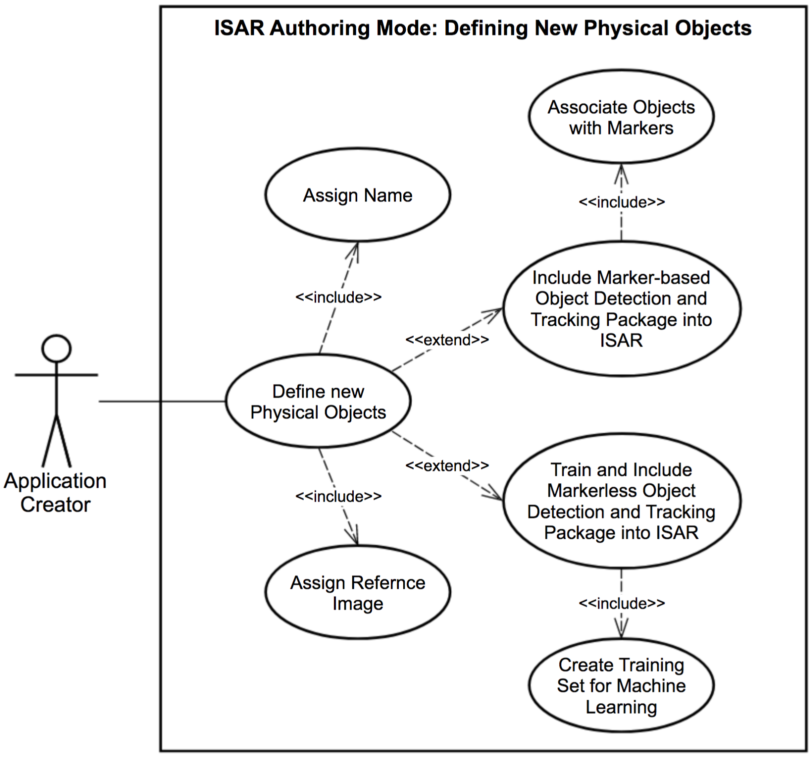

ISAR Authoring Mode: Defining New Physical Objects
Associate Objects with Markers
Assign Name
<<include>>
<<include>>
Include Marker-based Object Detection and Tracking Package into ISAR
<<extend>>
Define new Physical Objects
Application Creator
<<extend>>
<<include>>
Train and Include Markerless Object Detection and Tracking Package into ISAR
Assign Refernce Image
<<include>>
Create Training Set for Machine Learning

Fig. 3.4 In order to add new physical objects to ISAR, they must be first defined. To define a physical object it must be assigned a name, a reference image, and a marker-based or markerless object recognition and tracking model must be integrated into ISAR that recognizes this object.

3.3.2 Executing ISAR Applications

The application user executes ISAR applications in execution mode. An ISAR application consists of the description of the scenes, including physical objects and virtual annotations, the description of interactions, object recognition and tracking packages, and any media file needed for annotations, such as images, audio and videos. All the content of the application are packaged in a redistributable zip file.

In order to run the application, the application user opens the ISAR application in ISAR execution mode. In the runtime, the scenes are projected on the tabletops. The scenes are rendered and projected on the tabletop. The annotations are either rendered at fixed positions on the tabletop or, if attached to physical objects, are rendered based on the position and orientation of the object on the table.

The interactions of application user with the tabletop and projected annotations are governed by interaction rules defined by application author. The application user interacts with the table by manipulating objects and selecting different objects and annotations. Putting an object on the table, placing an object at a specific position, picking an object, moving the hand over an object are some of the examples of how the application user interacts with the objects on the table. Selecting annotations, e.g. a video, on the scene is another kind of interaction that happens with digital content. In additions, the passing of time itself can be seen as a kind of interaction with table. Based on the interaction rules defined by the application creator, the table responds to the interaction events with an action. For example, a workflow guidance application, the application creator might have defined a rule like "If pincers is picked at current step, give an audio warning hint". When the application user picks the pincers, the audio warning hint is played. Based on the interaction rules, an event might also trigger implicit transition to another scene.

3.3.3 Extending ISAR

The abstractions for designing the projected scenes and the interactions offer application creators the possibility to define the application content by themselves and without dependence on programmers. Yet, the set of abstractions offered by ISAR must be extensible in order to support new and unforeseen scenarios. To this end we define a third actor, the *framework extender*, whose job is to extent the functionality of ISAR. The framework extender is a programmer and can extend ISAR in three aspects: by defining new annotation types, by defining new event types, and by defining new action types that are used to define interaction

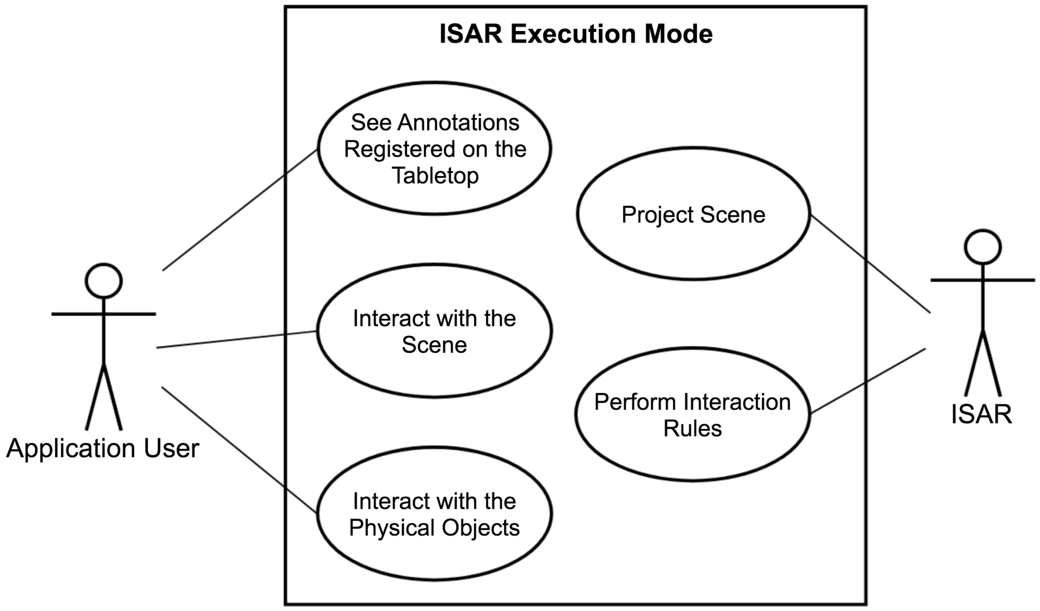

ISAR Execution Mode
See Annotations Registered on the Tabletop
Project Scene
Interact with the Scene
Perform Interaction Rules
Interact with the Physical Objects
Application User
ISAR

Fig. 3.5 The application user sees the scenes projected on the tabletop and interacts with the objects and digital content according to the interaction rules defined by the application author.

rules (figure 3.6). The new annotation, event, and action types can then be used by application creators to define scenes and interactions.

ISAR Framework offers base classes and functionality for annotations, events, and action and how to they are used to create scenes and interactions. The framework extender adds new annotation, event, and action types by inheriting from these classes and implementing the required functionalities and providing the required specification. Appendix C describes the details of extending ISAR framework.

3.4 Non-functional Requirements

ISAR is envisioned as an authoring and runtime system for real-time interactive applications that run on a tabletop. To achieve this goal, ISAR must fulfill the following non-functional requirements:

- **Authoring of Application Content for the Interactive Tabletop:** ISAR should allow the application creators to define the content and the interactions that make up an application that runs in real time on the interactive tabletop.

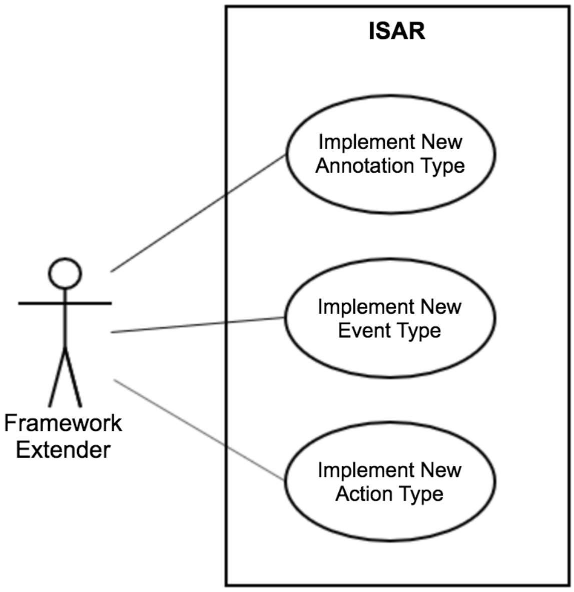


Fig. 3.6 The framework extender extends ISAR by adding new annotation types, new event types, and new action types to the framework.

- **WYSIWYG Authoring:** ISAR should provide the application creators with a WYSIWYG (What You See Is What You Get) authoring system. The definition of the scenes should exactly correspond to what is projected for the application user on the table in execution mode. The application creator should be able see the scene as it is being created and be able to test the interactions, object and hand tracking, and the situated rendering of the multimedia content during the authoring.
- **Rich set of Annotations:** ISAR's Authoring System should provide application creators with a rich set of annotations that can be used to define scenes for different scenarios. This should include simple geometric shapes like line, rectangle, circle, curve, and arrow; multi- media content like text, video, audio, and image; dynamic annotations like timer, counter, and animation; and interaction elements like touch buttons, checkboxes, object placement areas, and feedback annotation.
- **Definition of Interactions:** The application creator should be able to define how the application user interacts with the table and define the changes in the scene as the response to application user's actions.

- **End user Programming:** Defining the application content in authoring mode and using the application in the execution mode should not require any programming skills. The use of ISAR's Authoring System should be self-explanatory and simple to learn.

It should not require any previous knowledge about Augmented Reality and Interactive Tabletops and their underlying technologies. The step of the camera-projector system should be simple and an application creator without any technical background should be able to acquire the required camera and projector from consumer market and setup the system hardware.

- **Reusability and Adaption of ISAR Applications:** The application content created using ISAR should be easy to change an adapt for new scenarios. The application creators should be able to persist their projected created by ISAR's Authoring System, and reload the projects later for adaption and editing.
- **Scene Rendering and Projection of the Application Content on the Tabletop:** In the execution mode, ISAR's runtime system should render the application content for situated projection on the tabletop. The coordinate transformations between defined scene and tabletop scene should happen automatically and the annotations should be rendered according to position and orientation of physical objects on the tabletop.
- **Object Recognition and Tracking:** For situated rendering of the content on the tabletop, ISAR should recognize physical objects as defined by the application creator. ISAR should support both markerbased and markerless object detection and tracking. ISAR should have an accurate estimation of the position and orientation of tho objects on the tabletop with an error of maximum 10 pixels. The position of one hand on the tabletops should be tracked and determined as well with an error on maximum 10 pixels. Additionally a tracked selection stick should allow application users to interaction with the table, e.g. to select objects and annotations.
- **Extensibility:** ISAR should be extensible in two aspects. For creating the content for new scenarios, the application creator should be able to reuse already created applications and extend them by adding and adapting new scenes and adding new physical objects to the set of objects recognized by application. Adding new physical objects to ISAR should not require any technical background and ISAR should provide a simple GUI for the application creator to label new objects. For the marker-based object tracking, the application creator should be able to associate markers with physical objects. For the markerless tracking, the GUI should allow application creator to create an object detection dataset, and train an object detector that can be added to ISAR from the GUI.

  ISAR should also be extensible by a programmer to add new annotations and interaction types.

- **Performance:** Object detection and tracking and situated rendering of the scene should allow a frame-rate of at least 20 frames per second. The application users should not notice any delay in rendering of the scene on the table and response of the system to their interactions.

Chapter 4

ISAR Framework and Authoring Environment

In this chapter we describe the conceptual model of ISAR framework for creating interactive augmented reality tabletop applications by end user. ISAR Framework is designed to fulfill the requirements described in previous chapter. Following the conceptual model, we describe the system architecture and implementation details of ISAR and its two main application components ISAR Authoring Environment and ISAR Execution Environment.

ISAR has two separate usage modes: Authoring and Execution. In Authoring mode, the application creator designs the applications as a set scenes and interactions. In Execution mode the scenes are projected on the tabletop and the application user interacts with the application. ISAR targets application creators and application users with not technical background in computing, programming, and augmented reality. Simplicity towards the end user is one of design goals of ISAR. The hardware configuration for an interactive table based on ISAR is hence simple and consists only of a webcam and a consumer projector hanged on top of worktable.

4.1 ISAR Framework

The application content for an ISAR application are packaged in a project. A project is folder on the file system that contains all the files that are required for projection of application content on the table. This includes the description of the scenes, workflows, and interactions, as well as all media files included on the scenes like images, audio files, and video files.

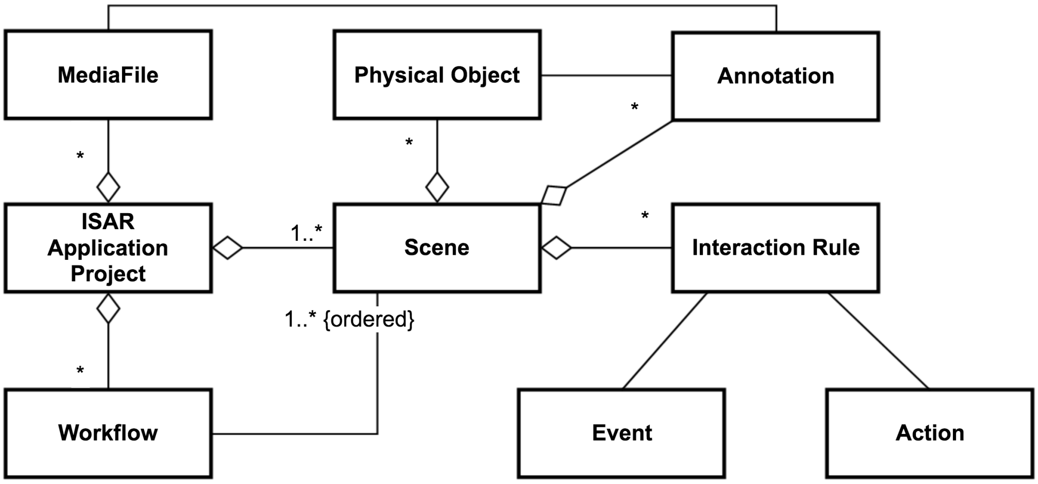


Fig. 4.1 Overview of Application Content Model in ISAR. Application content consists of a sequence of scenes consisting of multimedia annotation and physical objects. The interaction with the application is governed by interaction rules.

The UML class diagram in figure 4.1 shows an overview of ISAR's Application Content Model. The project is the container for the application content. The application is defined as a sequence of one or more scenes. Each scene consists of a set of annotations and physical objects. Annotations can be fixed on the scene, or attached to physical objects. In the latter case, the annotation is rendered on the scene based on position and orientations of the physical object. If the annotation is a multimedia annotation (image, audio, video) the corresponding media file is also contained in the project folder.

The application can contain a workflow. A workflow is defined as an ordered sequence of scenes, one scene for each step of the workflow. The project can also contain exercises. An exercise is also represented using a scene.

The interaction of the application user with the application and physical objects on the tabletop is defined by the application creator in interaction rules. Each scene can have a set of interaction rules. An interaction rule is a pair consisting of an event and action. The event signals a specific state of the interaction, such as 'physical object put on the table', and the action defines the response of the table, for example as a change in the scene.

4.1.1 Scene

Figure 4.2 shows an overview of the scene model in ISAR. A scene is a combination of physical objects and digital content, called annotations, that are projected on the table.

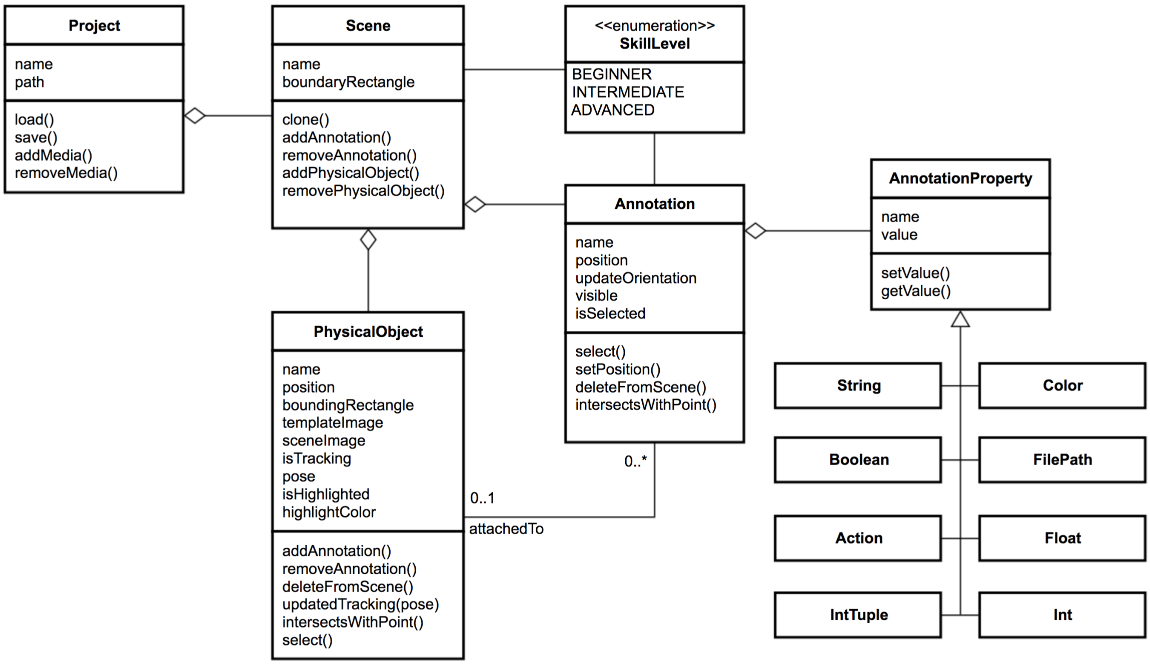


Fig. 4.2 Scene model: Annotations and physical objects are added to the scene. Annotations can be fixed on the scene or attached to physical objects. For extensibility AnnotationProperty is the super class for different annotation properties.

Application creator defines scenes by placing annotations and physical objects on the scene. When a physical object is added to the scene, its template image is shown on the scene in ISAR authoring environment. In the execution time, the object tracking component of ISAR determines the orientation of physical objects on the table with respect to this template image.

ISAR offers different kinds of annotations for defining the scenes (figure 4.3). Simple geometric annotations, including line, rectangle, ellipsis, arrow, and curve, can be used to design scenes. These geometric shapes can have different colors and thickness and can be filled or empty. The curve annotation can further be used for hand-eye coordination exercises. Multimedia annotations can be used to add text, image, audio, and video to the scene. Dynamic annotations include annotations that change their properties with time or as the result of user's interaction. These include timer, counter, relationship, and animation. The relationship annotations is a labled line connecting two physical objects and shows their relationship. The animation annotation can also be used for hand-eye coordination exercises. Finally, interaction annotation can be added to the scene for capturing user's interaction (see 4.1.2) or show feedback. Action button can be selected by the user to trigger an action. Object placement area triggers an object placement event whenever an object is placed in

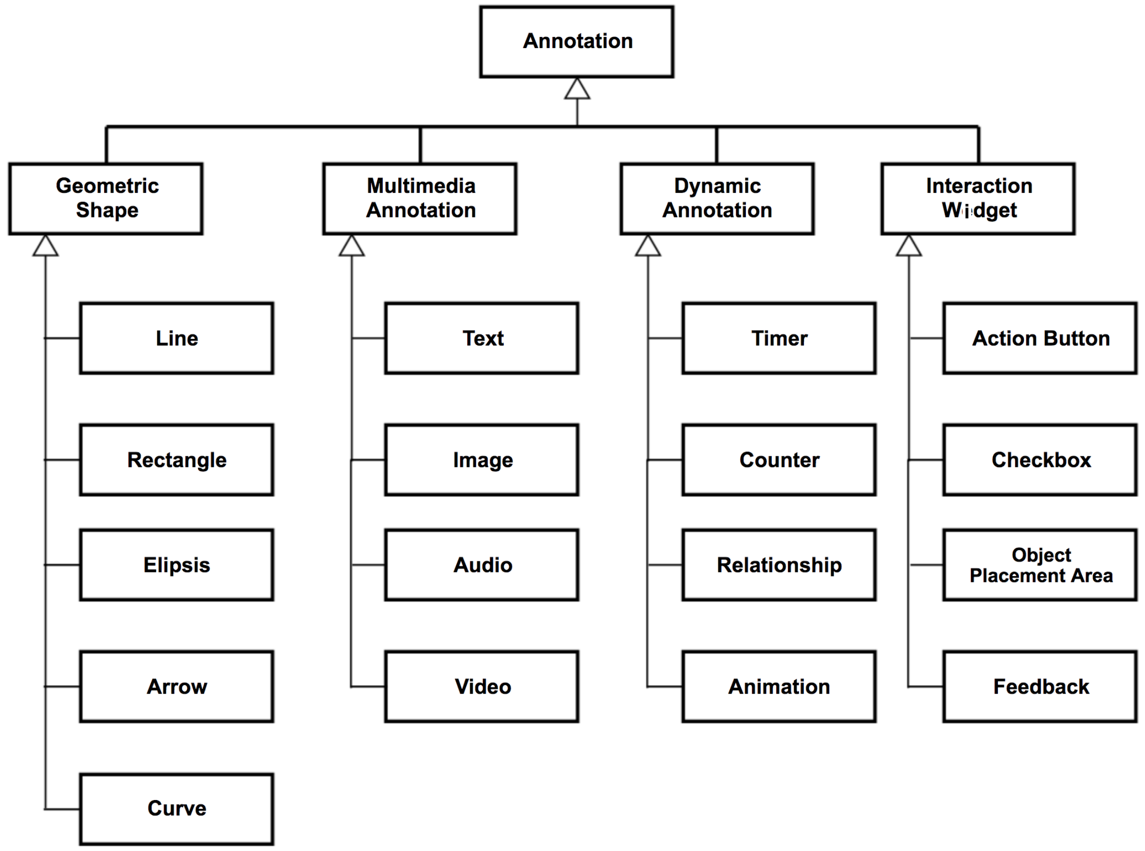


Fig. 4.3 Different kinds of annotations can be used to define scenes for various scenarios.

this area. The feedback annotation shows a feedback image, text, or plays a feedback audio based on interaction rules defined by application creator.

Annotations can be fixed on the scene, or attached to a physical object. If attached to a physical object, the position and orientation of the annotation are determined in the object's coordinate system. Each annotation has a set of properties. Position, name, and visibility are common to all annotations. Other properties are based on type of annotation, for example an image annotation has a size and and image path. Appendix A lists all the annotations in current version of ISAR, their properties, and how they are rendered on the tabletop.

One of the main application domains for ISAR application are educational applications. The purpose of an educational application is to support improvement of student skills. Accordingly, ISAR supports the concept of skills. Each annotation can be associated with a skill level. There are three skill levels: beginner, intermediate, and advanced. In the execution mode, the application user can select the skill level. In that case, only annotations corresponding to that skill level are shown.The skill level can also be defined for an entire

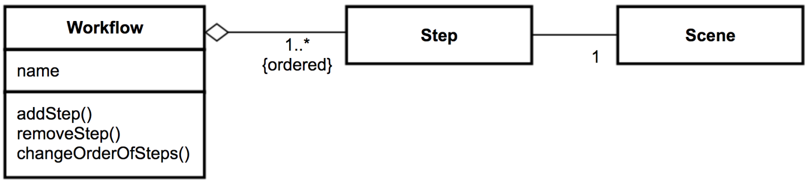


Fig. 4.4 A workflow is an ordered sequence of steps, each associated with a scene.

scene. In that case it is applied to all annotation of that scene. For example a scene for a beginner skill level might have more detailed instructions compared to a scene for intermediate user. By associating skill levels to entire scenes, it is also possible to define skill-oriented workflows. The sequence of scenes constituting a workflow can be different based on the selected skills.

Workflows

An ISAR application can contain one or more workflows. A workflow is a sequence of scenes, one scene for each workflow step. The workflow is defined by ordering the scene in a navigation flow that determines the start scene, end scene, and the ordering of the scenes. Each step of the workflow is associated with one scene. The transition between the steps happens either explicitly, for example by selecting a 'next' button, or implicitly based on interaction (see interaction model 4.1.2), for example as the result of placing a physical object at a designated position.

The standard navigation flow for the scenes of a workflow determines the next and previous scenes. It is also possible to define scenes that are not part of the navigation workflow, for example a help scene that shows help for specific step of the workflow. These scenes must be shown explicitly. At the execution mode, the execution environment of ISAR keeps track of the scenes as they appear. This allows going back and forward between scenes, in addition to the standard navigation flow.

Exercises

Having educational applications in mind, it is also possible to create different kind of exercises (figure 4.5) by combining annotations and interaction rules (4.1.2). For example the checkbox annotations can be used to create multiple choice exercises that are projected on the table. The multiple choice exercises can also have physical objects as answer, because checkbox annotations can be attached to physical objects. It is also possible to create exercise

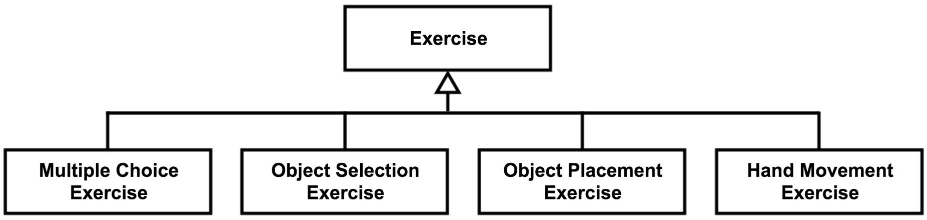


Fig. 4.5 Different kinds of exercises that can be defined in ISAR by combining different annotations and interaction rules.

that involve selecting objects by defining interactions rules or exercises for placing objects at a specific position by using an object placement area annotation and configuring the corresponding interaction rules.

A specific kind of exercise supported by ISAR are hand-eye coordination exercise. Using curve annotation and animation annotation it is possible to create exercises that training of motor skills, for example by following a projected path on the tabletop.

4.1.2 Interaction

In order to create and interactive application, the application creator must be able to specify the interaction of the application user with the objects on the tabletop and the with the projected scene. Interaction rules allow specification of interactions without the need for programming. An interaction rule is a pair consisting of an event and an action (figure 4.6). When the event of a rule is detected, the action is triggered. An event can come from different sources, such as annotations, scene navigation, time, or manipulation of the the physical objects. An action in turn can change the state of the scene, annotations, or physical objects.

Events are triggered as the application user interacts with the physical objects on the tabletop or with the projected annotations (figure 4.7). Event sources trigger events. An event source can be a scene, for example when the scene appears or disappears, annotation, for example when an annotation is selected, or physical objects, for example when physical object is placed on the table. Passing of the time is also another source of events that is triggered by timer annotations. Each annotation has a set of events that can be fired when the state of annotation changes. Figure 4.8 shows the different kinds of event in current version of ISAR. The application creator configures an event in ISAR's Authoring Environment by selecting its type, it source, and configuring values for event properties, if any.

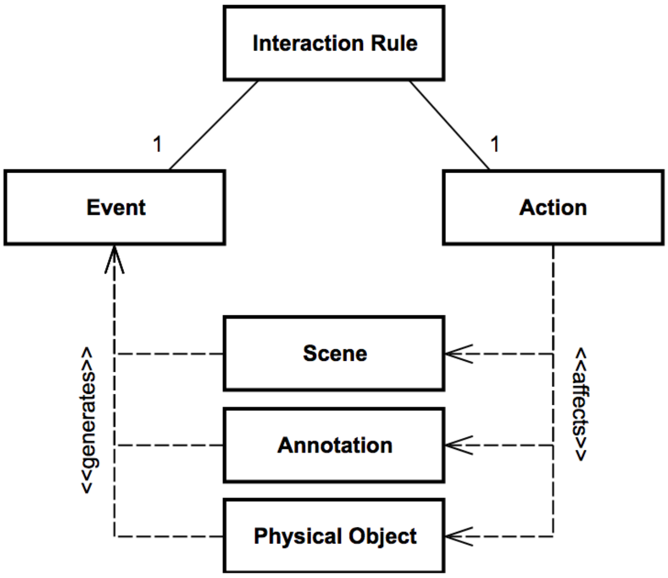


Fig. 4.6 Overview of the interaction model in ISAR. The application creator defines an InteractionRule as a pair of Event-Action. As the application user interacts with the tables, e.g. by manipulating physical objects, Events are triggered. If the triggered event matches the event of an interaction rule, the Action of that rule is performed, which affects the state of the table, for example toggling visibility of an annotation or playing a sound.

Events are triggered either by observer threads, for example the hand tracking service triggers an event whenever the hand is over an object, or by annotations, for example the timer annotation triggers an event when a defined timeout is reached.

Actions are used to define the system's response to user's interactions, for example by changing the state of the scene. Figure 4.9 shows different kinds of actions available in current version of ISAR. There are actions for scene navigation, changing the visibility of annotations, starting and stopping timers, videos, and audios, and highlighting physical objects or specific parts of the scene. Composite actions allow combining several actions into one. Composite actions can either be parallel composite action, or sequential composite action. For a parallel composite actions, all the sub-actions are performed in parallel, when the composite action is to be performed. For a sequential composite action, the sub-actions are performed one after another in the defined order.

An action can have a target. This is the scene, annotation, or physical object whose state is changed by the action. Similar to events, the application creator configures actions in ISAR's Authoring environment by defining action type, action's target, and actions properties.

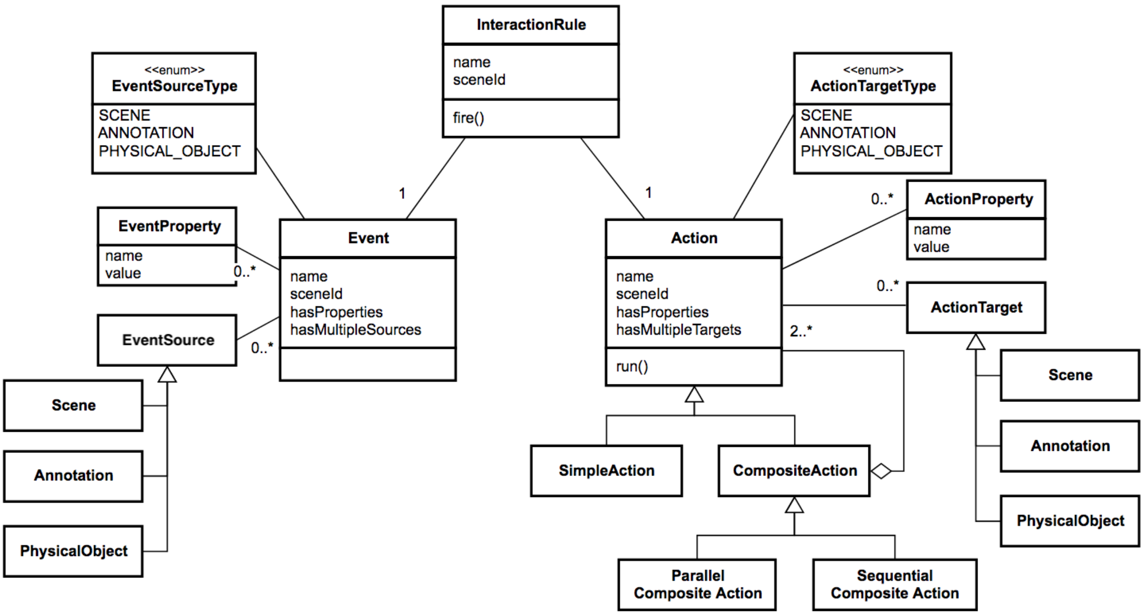


Fig. 4.7 Interaction model classes in details. Event sources trigger events. Actions affect targets. Events and Actions can have properties for their parameters.

In execution mode of ISAR, an observer thread listens for all the different event types that are triggered. When an event is triggered, it is compared with the event part of all the define rules. If there is any rule whose event part matches the triggered event, it is fired; that is, its action is performed. In addition to being performed when a rule is fired, the actions can also exist as stand-alone actions that are performed explicitly by user. For example an action button can be used to perform a scene transition action.

4.1.3 Object Detection and Tracking

Rendering of annotations on the tabletop and different interactions require tracking the position and orientation of the physical objects. Other interactions like selection of the objects or annotations on the table, and picking and moving of the objects require tracking the position of a selection stick and the hand. ISAR provides markerless and marker-based tracking of the objects and marker-based tracking of a selection stick and the hand.

Scene annotations that are attached to a physical object must be rendered according to the position and orientation of the object on the table. Object detection and tracking is used to determine the position and orientation of the objects. We use the term *object pose* to refer to the combination of position and orientation. Beside rendering of the annotations,

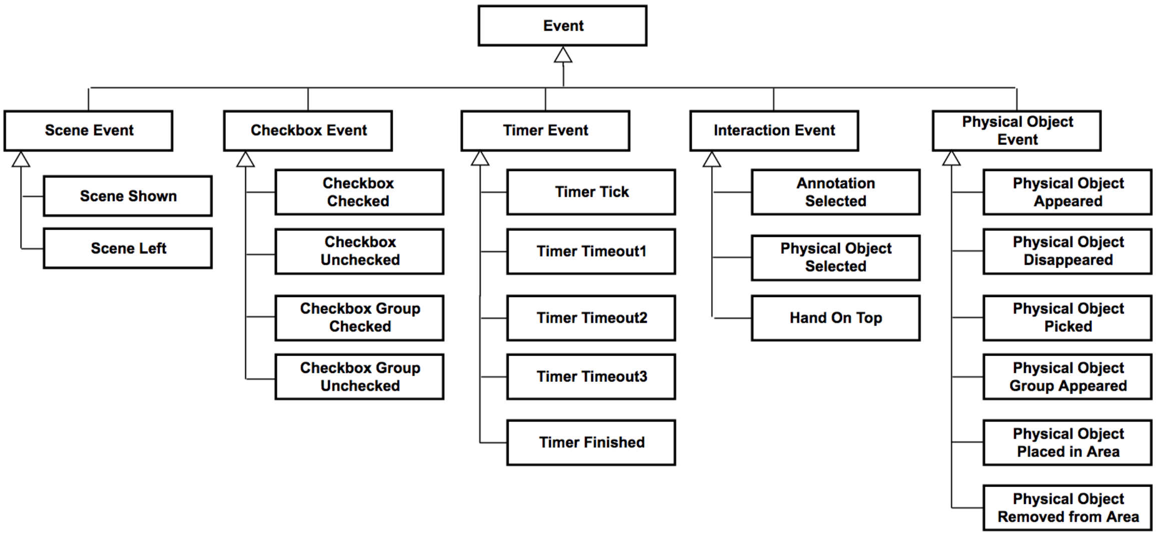


Fig. 4.8 Different kinds of events in ISAR. Events are triggered for Annotations, Physical Objects, or Scene.

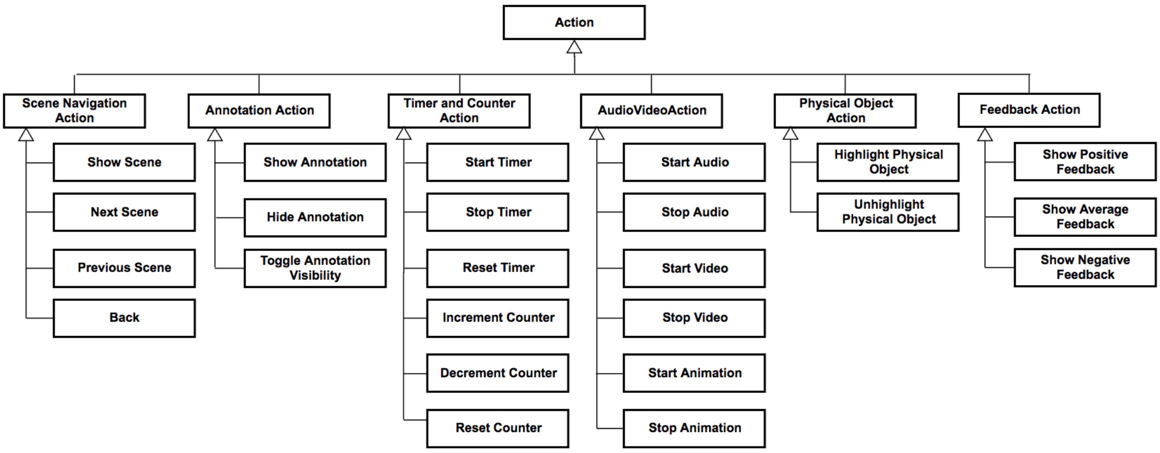


Fig. 4.9 Different kinds of actions in ISAR. Action can have Annotation, Physical Object, or Scene as target.

different interaction events related to manipulation of objects, such as picking and object or placing an object in a specific area on the table, or removing an object from the table, require object detection. In order to create their own applications, end users must be able to create their own object detection and tracking packages and integrate them into ISAR. We took a plugin approach for extending ISAR with new object detection packages. ISAR supports both marker-less and marker-based object detection and tracking. For both approaches we provide a GUI for the user to automatically create the object detection package that is then integrated into ISAR's plugins folder or into their application package.

Figure 4.10 shows the object detection model of ISAR. The object detection service provides a single point of access to all available object detection packages. When the service starts, it searches the object detection plugin path of ISAR authoring or ISAR runtime application to find the existing object detection packages. An object detection package must provide at least two things:

- An object detector descriptor: This is a Pyhthon module with the name *objectdetector.py*. The object detector module must provide the description of all different physical objects that can be detected by this module. This list includes the names of physical objects and the path to their corresponding template image.
- A folder containing the template image for each of the detected objects.

For each object detector found in the plugins folder an ObjectDetectionWorker process is created. This process loads the actual object detector module and forwards requests for object detection to it. Each object detection worker process has also a corresponding observer thread that waits for the object detection results to become ready and calls an asynchronous callback with the results. ObjectDetectionPrediction objects contain the result of object detection. Each object contains the name or label of the detected objects as well as their position and orientation.

This design fulfills the requirements of extensibility and realtime performance (see 4.3.1). By adding new object detection packages new physical objects can be used to design scenes in an ISAR application. Object detection works independent of how internally the object detection package detects objects and estimates their pose. Furthermore, the asynchronous call to object detectors allows continuous updating of object detection information regardless of how long it takes for different object detectors to return the position and orientation of their corresponding objects. In this way object detection is not blocked by the slowest object detector.

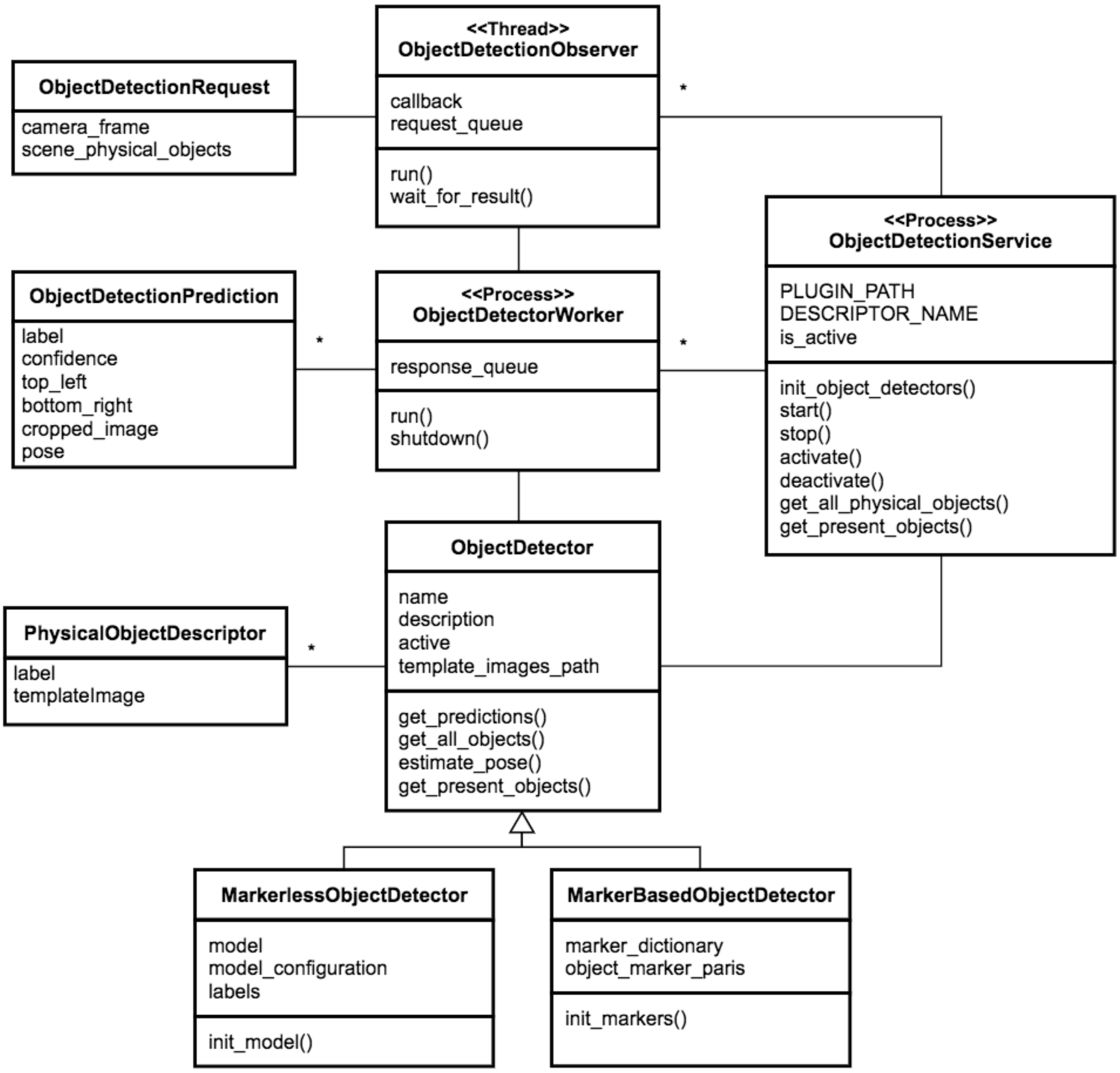


Fig. 4.10 Object detection model of ISAR. New object detection packages can be add as plugins by dropping them into the respective plugin folder or by integrating them into ISAR application package.

ISAR provides a GUI tool for creation of both marker-based and markerless object detection packages. The markerless object detection relies on YOLO v3 [RF17] CNN model and 2D euclidean pose estimation based on AKAZE [AS11] feature descriptors. Details of the pose estimation method and the object detection network can be found in appendix B.

The GUI and a server process allow user to create the dataset for training the machine learning model for markerless object detection. The user puts the objects on the tabletop and sees them in a camera stream, that he can pause. When the camera stream is paused, bounding boxes of all them objects are automatically detected and the user can assign labels to each bounding box. This creates a training sample. The user only needs to create few samples and system uses data augmentation techniques to create a training dataset. The training dataset is then sent to the sever and training of the model starts. When the adequate accuracy is reached, the users gets a notification and can download the complete object detection package including the trained model and integrate it into ISAR.

For creation of the marker-based object detection packages, ISAR provides a simple GUI that allows user to associate markers with objects. Marker-based object detectors use ArUco markers [GJMSMCMJ14] and the corresponding tracking library. The markers must first be attached to the objects. The users put the objects with the markers attached on the table. In the video stream of the tabletop, then user can select the bounding box or each object and assign it a label. The advantage of the marker-based approach is its simplicity and fast creation of object detection packages. The disadvantage of this approach is that exactly the same marker must be attached to the objects in the execution mode.

In addition to object detection and tracking, ISAR provides hand tracking and interaction using a tracked selection stick. For both hand tracking and selection sticks we use ArUco markers. For hand tracking a marker should be attached to a glove that must be worn by the application user in execution mode. A marker is also attached to the tip of the selection stick. The application user or application creator can configure in the setting which marker is attached to selection stick and glove. In spite of the obtrusiveness of the markers, we decided to use marker-based tracking for hand and selection stick interaction because of its simplicity, speed, and reliability.

4.1.4 ISAR Framework as a Meta-model

ISAR Framework provides abstractions and functionality that is used to create and run interactive tabletop applications by non-technical users. These abstractions and functionalities

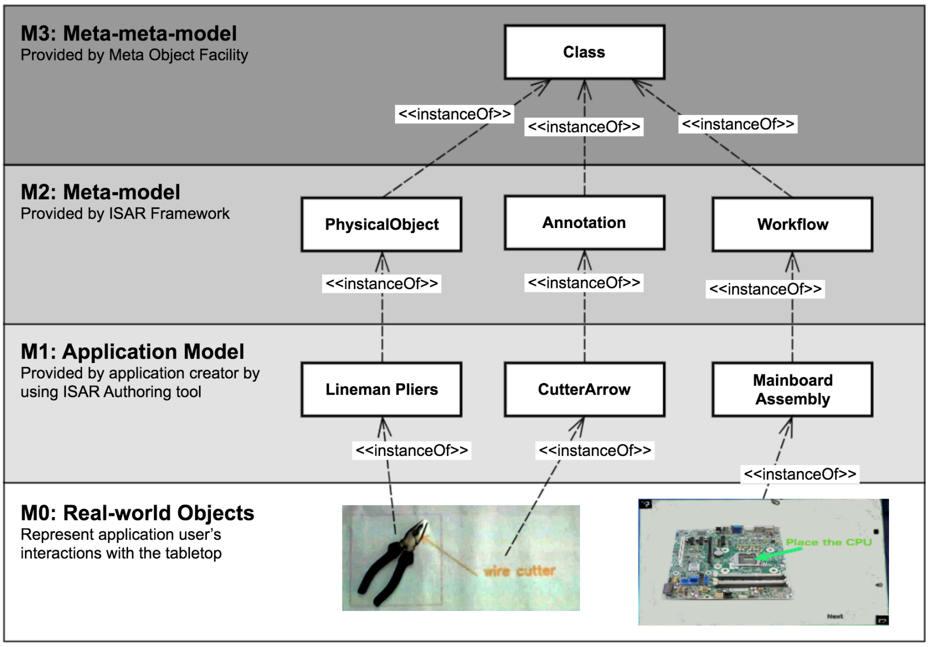


Fig. 4.11 ISAR Framework can be considered a meta-model at level M2 of the Meta Object Facility. This meta model is used by application creator to create an ISAR application for a specific scenario.

are provided to the application creator through the authoring environment. In this way, we can consider ISAR Framework as a meta-model that describes the models created by the end user. The Meta Object Facility [Poe06] by Object Management Group (OMG) provides a formalism for describing such meta-models. MOF consists of four layers:

- **Layer M3**: At the highest level the constructs of meta-meat-model, such as *Class*, are provided by the Meta Object Facility
- **Layer M2**: The constructs from layer M3 are used in layer M2 to create a meta-model. A meta-model is a model that describes a model. Frameworks are often defined at this level.
- **Layer M1**: At this layer the framework users use the constructs provided by the meta-model to create their own models for their corresponding scenarios.
- **Layer M0**: This layer finally corresponds to the real-world, where the instances of the user's defined model are created and worked with in the application.

Figure 4.11 shows the relationship of ISAR Framework to Meta Object Facility. ISAR Framework is a meta-model at layer M2 of MOF. The abstractions provided by the ISAR Framework are instantiated and used by the application creator in order to created ISAR applications. ISAR applications can in turn be considered as models of the specific problem scenario that the application creator needs to address. Finally, at the execution time, an ISAR application is instantiated and the application user interacts with real instances of physical objects and virtual information in form of scenes projected on the tabletop.

4.2 Authoring and Execution Environment

ISAR consists of a camera-projector setup, an authoring system for creating applications, an execution environment for running applications, and the utilities for creating object recognition and tracking packages. Figure 4.12 shows the hardware configuration of the system. The hardware configuration consists of a simple webcam and a consumer mini or pico projector, that must be hanged on top of a table. The end user is free in the choice of the webcam and the projector as long as they both support at least HD resolution. The projector and the camera should be placed on top of the table in a way that the projection area of the projector is visible in field of view of camera. The background of the table should be of white color for better detection of the objects and visibility of the projected information. Two markers at top-left and bottom-right define the boundaries of the scene on the tabletop. All coordinates of the scene are defined with reference with the (0, 0) point of the scene at the top-left corner of the scene. In the authoring application the camera image is cropped so that only the scene boundary is visible for defining the scene. In the projection mode, a rectangle is projected around the scene boundaries for better orientation of the application user.

Figure 4.13 shows the ISAR Authoring environment and its different parts. The authoring mode provides a WYSIWYG (What You See Is What You Get) environment for the application creator. Application creator can see the scenes exactly as they would be projected on the table in the authoring mode. Defined interactions and object tracking can also be tested exactly as they would be used in the execution mode. The application creator see the camera stream from the camera on top of the table in the scene viewer (1). The camera image is cropped at the scene boundaries defined by the two markers. In the scene list area (2 and 3) application creator can see a list of scenes, add new scenes, delete scenes, or change the order of the scenes and define a default navigation flow. Annotations can be added to the scene by using annotation tools (4). The application creator activates an annotation tool and uses the mouse two draw the annotation on the scene. Annotation can also be selected and deleted and changes to the scene can be undone or redone (5). Selecting an annotation on the scene or on the list of annotations (6) shows its properties in the properties view (7). All annotations types have a set of common properties such as position and attach_to, that

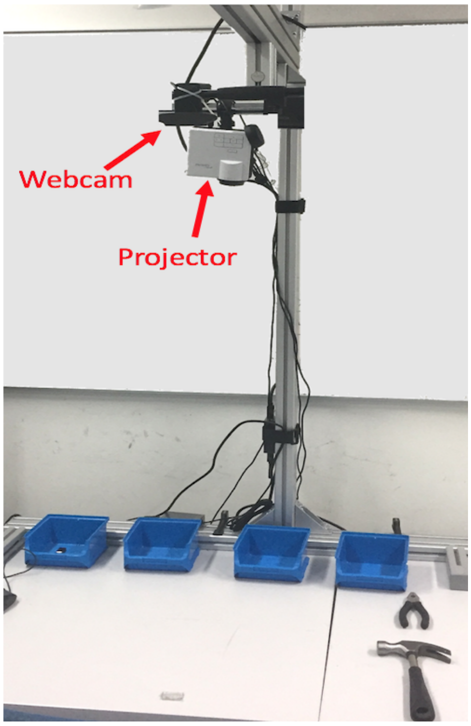
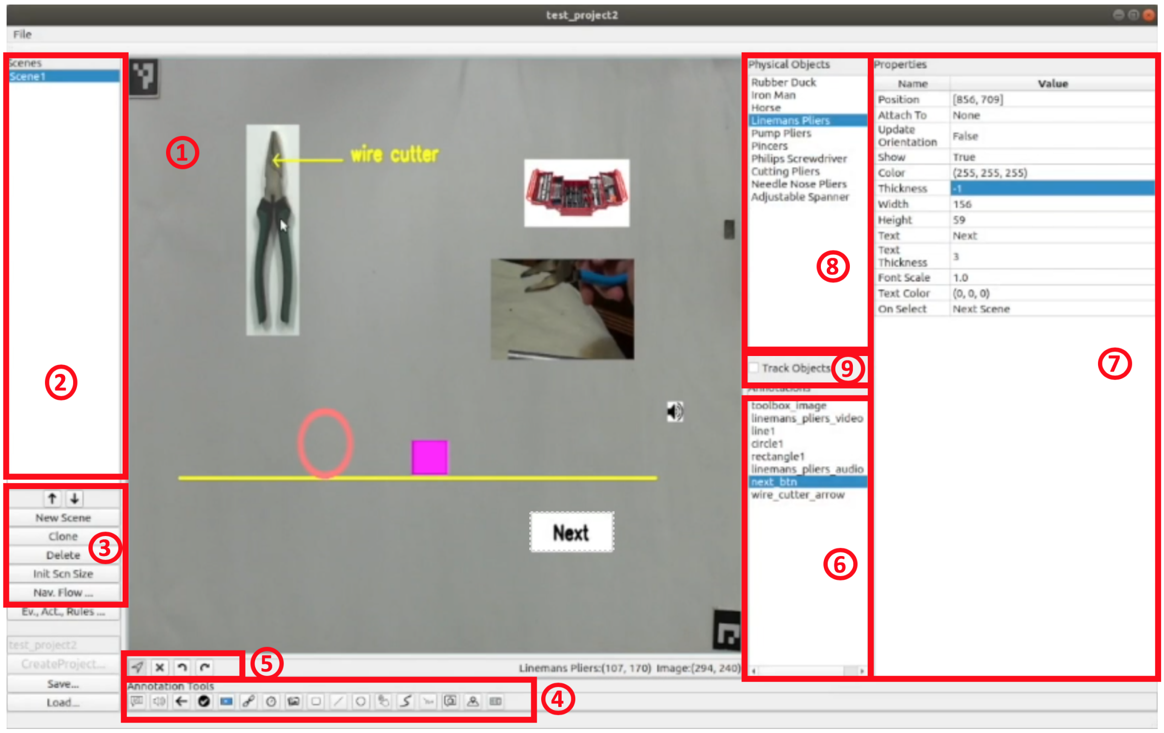


Fig. 4.12 Camera-Projector setup. Using a projector and a webcam, every ordinary tabletop can be used for ISAR applications.

define the position of the annotations and if it should be rendered with respect to a physical object. In addition to that, each annotation type has a different set of properties that all can be dynamically changed in the properties view. The physical objects view (8) shows a list of all physical objects that can be recognized and tracked based on existing installed object recognition and tracking packages. The application creator adds a physical object to the scene by dragging and dropping it on the scene. Dropping a physical object on the scene renders its template image. If annotation is attached to the physical object, its position is with reference to the template image. A coordinate viewer at the bottom right of the scene viewer shows the coordinate of the mouse cursor both with respect to the physical object and the scene. If object tracking is activated (9) and the scene physical objects are put on the table, the object is tracked instead of its template image being shown. dgfm. In this case all annotations attached to the objects are also rendered according to their position and orientation. In this way, the application creator can directly test the application from the authoring environment.

Different annotations can be added to the scene. For example the scene in figure 4.13 shows an arrows annotation attached to the physical object lineman pliers, an image annotation showing a toolbox, a video annotation showing who to use a lineman pliers, an audio



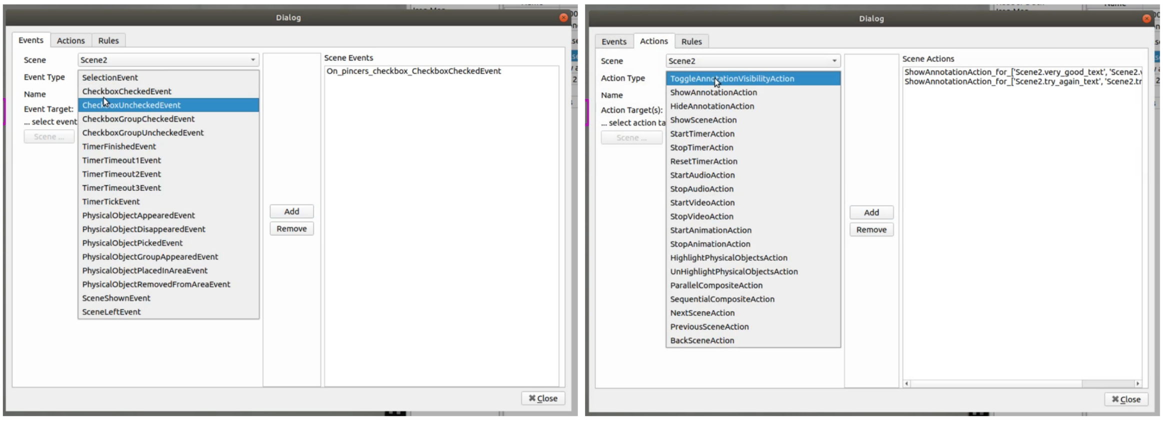
Name	Value
Position	[856, 709]
Attach To	None
Update Orientation	False
Show	True
Color	(255, 255, 255)
Thickness	-1
Width	156
Height	59
Text	Next
Text Thickness	3
Font Scale	1.0
Text Color	(0, 0, 0)
On Select	Next Scene

Linemans Pliers:(107, 170) Image:(294, 240)

Fig. 4.13 The WYSIWYG authoring environment of ISAR. In the authoring mode, the application creator sees the scenes exactly as they would be projected at execution mode.

annotation that pronounces the name lineman pliers, geometric shapes line, circle, rectangle, and an action button that changes the scene when selected.

The application creator adds each annotation by using the corresponding annotation tool from toolbar. Annotation tool classes are also responsible for rendering annotations. In drawing mode, annotation tool responds to mouse interactions of application user to draw the annotation on the scene. For example for drawing a rectangle, application creator select the rectangle annotation tool, then clicks on the desired position on the scene and drags the mouse to desired size of the rectangle. When the user releases the mouse, a rectangle annotation with the corresponding size is added to scene at the corresponding position. The application creator can now configure other properties of this annotation such as its color and line thickness, and including its position and size. For the annotations that need a media file, such as image, audio, and video annotations, the application creator can select the media file from file system. The media file is then copied to the project folder and is used for rendering the annotation on the scene. When the application is packaged all media files are also contained inside the application package.



On_pincers_checkbox_CheckboxCheckedEvent

ShowAnnotationAction_for_['Scene2.very_good_text', 'Scene2.v
ShowAnnotationAction_for_['Scene2.try_again_text', 'Scene2.tr

Fig. 4.14 Interaction rules are defined as event-action pairs. The application creators uses the interaction dialog to configure the events and actions that are then used to create interaction rules.

Figure 4.14 shows the dialog for defining interaction rules. An interaction rule is defined as an event-action pair. ISAR provides a set of event types that correspond to different states of the scenes, annotations, and physical objects. The application creator selects an event type and configures its parameters. For example for a *SelectionEvent* that application creator selects which object or annotation is the target of selection event. ISAR also provides several action types that can change the state of the scenes and annotations. Similarly, application creator selects an action type and configures its parameters. For example for a *HighlightPhysicalObjectAction* the application creator selects which physical object should be highlighted and with which color or for *StartAudioAnnotation* he configures which audio annotation on the scene must be played. Finally, after defining and configuring events and actions, the application creator creates interaction rules as event-action pairs. For each interaction rule the application creator selects which event should trigger which action. For example a rule can be like "Upon *SelectionEvent on Lineman Pliers* do *StartAudioAction for LinemanPliersName*", where a physical object with the name Lineman Pliers and an audio annotation with the name LinemanPliersName must exist on the scene.

The event detection service observes all the events that happen on the table and when an event of the defined type and configuration happens, it triggers the event. Consequently the interactions rules service gets notified of the event and checks among all the interaction rules, if there is a rule with the given event as premise. If such a rule is found its corresponding action is executed.

The application creator can directly test the interactions from authoring environment. For example the application creator may create a scene that contain a physical object and an

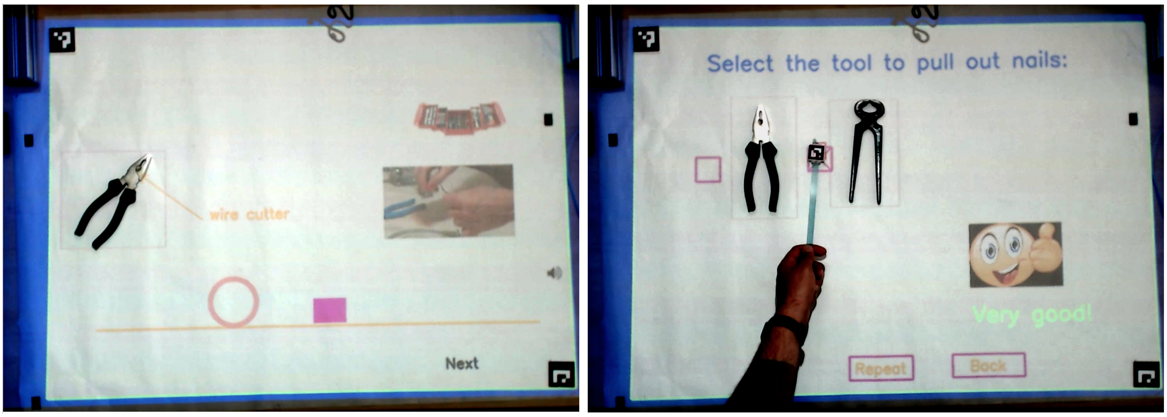


Fig. 4.15 In execution mode the application content is projected on the tabletop. Physical objects are tracked and annotations attached to them are rendered according to their position and orientation. The interactions of the application user and response of the system is governed by the defined interaction rules.

audio annotation for saying the name of that object, and an interaction rule like "if object selected, play the audio annotation". Right after defining the scene and the interaction rule, the application creator can put the object on the table, enable object tracking in authoring mode, and point to the object using the selection stick. This interaction will then play the audio annotation. In this way the application creator can test and adapt the scene and interaction right at the same time and environment as he is designing the scene.

At any time during designing the application the application creator can save the project. This saves the application structure, including all the scenes, their contained annotations and physical object, and all the interaction rules into a JSON file. After all the scenes and interactions are designed, the ISAR application can be saved and packaged for delivery to application user that uses the execution environment for running the application. The application package contains the project file describing all the scenes and interactions, all the media files needed for multimedia annotations, and the object recognition and tracking packages that were used during authoring.

ISAR's Execution environment is used to open and run an ISAR application. The execution environment consists of two views. A small view in which the application user opens the ISAR applications, can use mouse to select different scenes directly, can calibrate the camera and projector, and can configure different settings. The application itself runs in the projector view which renders the application content to be projected on the tabletop (figure 4.15).

The projector view in execution environment and the scene viewer in the authoring environment both use the scene renderer component to render the scene. The rendered scene

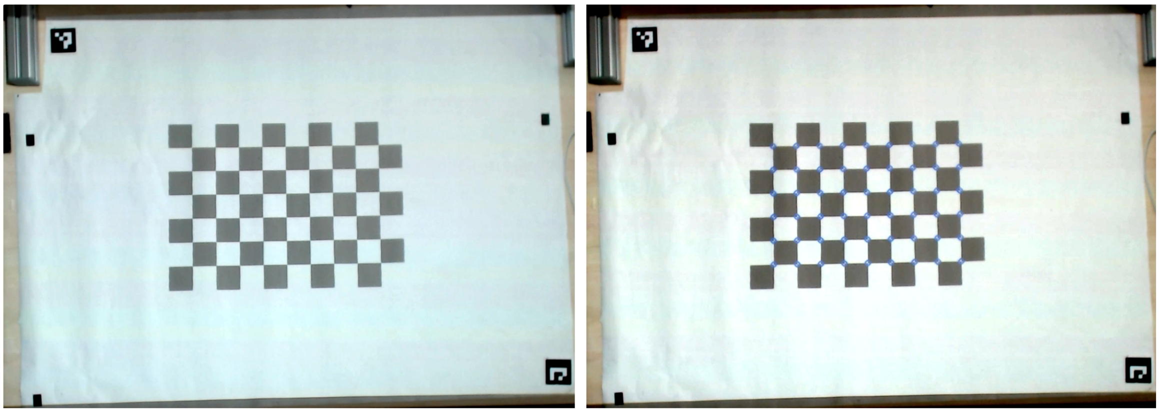


Fig. 4.16 To determine the transformation between the camera and the projector coordinates a known chessboard pattern is projected on the tabletop and the transformation is determined from set of correspondence points. The image on the right shows the reprojected chessboard points designated using circles. The markers for scene boundaries can be seen as well.

is an image of the scene boundaries on which all the annotations and graphical elements of the scene are drawn. In the authoring mode, this rendered scene is then overlapped with the camera image and shown in scene viewer. In the execution mode, the rendered scene is projected on the tabletop. This design allows for the WYSIWYG authoring, in which exactly the same rendered scene is either shown in scene viewer or is projected on the tabletop.

Several coordinated transformations must happen in order to render the scene and show it in the scene viewer and projector view. The scene boundaries are specified by two markers at the top left and bottom right corners of the scene on the tabletop. In the authoring mode, the image obtained from camera is cropped to the scene boundaries. This cropped image is the scene image on which all the annotations are rendered and which is shown in the scene viewer. All positions, for physical objects, annotations that are fixed on the scene, the tracked selection stick and glove, are with reference to (0, 0) of the cropped scene image. The scene image is resized to the size of scene viewer in the authoring window. For correct positioning of the annotations when they are added to the scene by mouse interaction, the mouse coordinates on the scene viewer must be transformed to scene coordinates. This is achieved by multiplying the mouse coordinates with the ratio between the size of the scene image and size of the scene viewer.

In execution mode all coordinates must be transformed to projector coordinates. For this transformation, first the scene boundaries in the coordinates systems of the projector must be determined. The scene boundaries are determined with the markers that are seen by the camera and in order to transform them into the coordinate system of the projector we need the transformation between camera and projector coordinates. This is achieved by camera-projector calibration. The projector projects a known chessboard pattern on the tabletop. This image is seen by the camera and the chessboard corners are determined. This gives us a set of correspondence points between projector and camera coordinates using which we calculated the transformation (figure 4.16). Since both camera and projector images are planar, the transformation is a homography.

The camera-projector homography allows us to transform camera coordinates to projector coordinates. Using the coordinates of the two scene boundary markers at the corners of the scene, the coordinates of the scene boundaries in the projector image are determined. This is the scene image in projector coordinates. Using the camera-projector homography all the scene coordinates of annotations are transformed to the projector coordinates and annotations are rendered on the scene image. The scene image is then placed inside the projector image that is projected on the tabletop.

The annotations that are attached to physical objects are rendered with reference to the coordinate system and orientation of the physical object. The (0, 0) point of the coordinate system of a physical object is the upper left corner of its bounding box returned from object recognition and tracking component. The position of the annotation is with respect to this origin point. All other coordinates of the annotations, for example end points of a line annotation, are then determined based on the affine transform of the object's pose calculated by the object detection and tracking component.

4.3 Architecture

This section presents the design goals and the proposed architecture of ISAR. Design goals are extracted based on the non-functional requirements from chapter 3 and justify the design decisions for the proposed architecture. The architecture of ISAR describes its different components and how these components interact with each other.

4.3.1 Design Goals

Design of ISAR is based on the following design goals:

- **End user Programming:** ISAR is motivated by enabling end users with no technical expertise in programming and augmented reality, to create their own augmented reality interactive tabletop application without the need to consult a programmer. ISAR should provide an authoring systems for creation of interactive tabletop application with no programming. Furthermore to address the usability for non-technical users, the authoring system should allow users to see and test the application as they are creating it (WYSIWYG). ISAR should support creation of interactive tabletop application for different scenarios, such as educational applications. To this end, a rich set of annotations should be offered to the end user to design the projected scenes that combine digital content with physical objects. In addition to designing the scenes, the interactions should also be defined by the user without programming. Furthermore, to lower the entrance barrier of non-technical users, ISAR should have a simple low-cost hardware configuration.

- **Domain-independence:** ISAR should support interactive tabletop applications for different application domains, such as educations, manufacturing, entertainment, and medical rehabilitation. In order to create augmented reality interactive tabletop applications that combine digital content with physical objects, the end users should be able to created object detection and tracking packages on their own and integrate them into ISAR. Furthermore, an application that is already created should be reusable and adoptable to new scenarios and event new domains. The users must be able to save and share their applications and adopt them to new scenarios with minimal effort.

- **Situated Projection of Application Content:** The application content should be projected on the tabletop exactly as the application creator designed it. The physical objects on the table should be recognized and their presence, position, and orientation should be tracked. The digital content attached to physical objects should be rendered according to position and orientation of the corresponding physical object. Hand tracking and a tracked selection stick should allow physical interaction and direct manipulation of the application content on the tabletop. Object recognition and tracking as well as hand tracking and tracking of the selection stick should be robust and accurate to allow uninterrupted natural interaction with application content. The performance for rendering of the scenes and response of the system to user's interactions should allow smooth real-time interaction of the user and the application.

- **Extensibility:** The ISAR framework and the authoring and execution environments should be extensible. A programmer should be able to add new annotations for designing the scenes, and add new event and action types for defining new interactions. This is in addition to reusability and adaptability of ISAR applications created by the end user. This design goal addresses the need for extending the framework and authoring

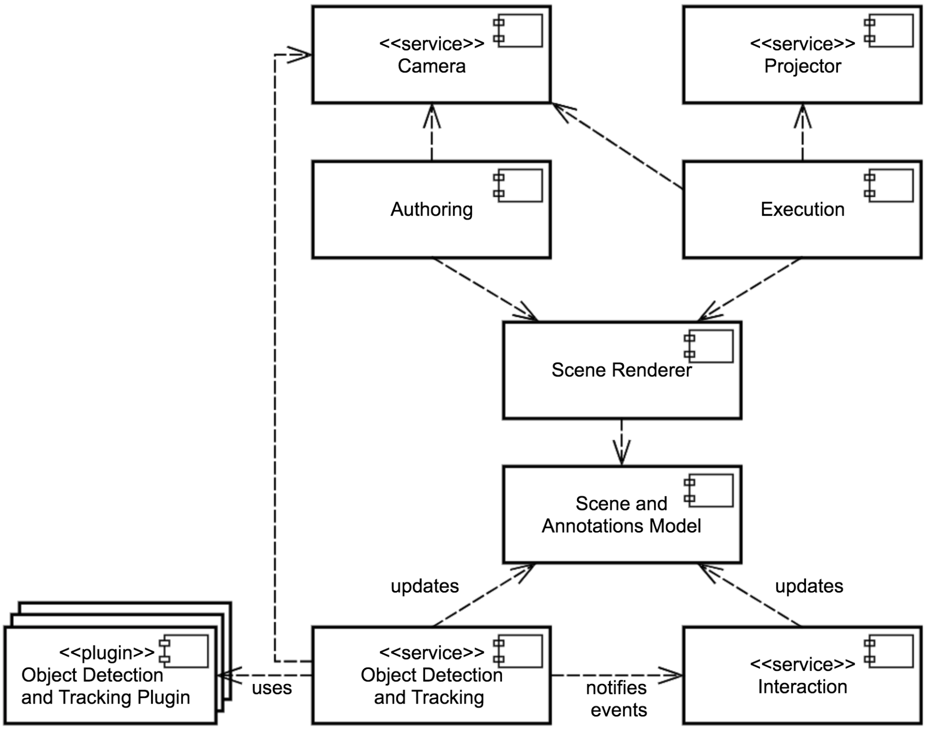


Fig. 4.17 ISAR subsystems. Authoring and Execution environments both depend on Scene Renderer to present the scenes and annotations. The state of the scenes and annotations are updated by Object Detection and Tracking component and the Interaction component.

and execution environments to support new scenarios and application domains not foreseen in the current version of ISAR.

4.3.2 Subsystem Decomposition

The architecture of ISAR is based on the repository architectural style [BD03] with the scene as the central repository concept that is updated and queried by other components. Furthermore, in order to fulfill the design goals of real-time performance and extensibility, we took a service-oriented approach to support flexible and fast communication between loosely coupled service components. Different components run in separate processes and offer interfaces for both synchronous and asynchronous service calls. A service registry manages the unified startup, shutdown, and access to the services.

Figure 4.17 shows the main components of ISAR and how they depend on each other. In both authoring and execution mode application runs in the main UI loop that coordinates the communication between services by passing the data to services and rendering the scene based on the results. Each cycle starts by getting the image of the tabletop from camera service, passing it to object detection and tracking and getting the pose of the existing objects on the table, and rendering the annotations. In authoring mode the annotations are rendered on the camera image which is then shown in the scene viewer. In execution mode, the annotations are rendered on an empty scene image that is projected on the tabletop.

Leaning on repository pattern [BD03] the Scene and Annotations component is the repository instance that services update or query. For example the object detection service updated position and pose of the PhysicalObject instances that are present on the scene, or the interaction service updated the visibility state of annotation or issues scene transitions based on user's interactions. The scene render in turn queries the current scene and renders its annotations on the scene image which is shown in scene viewer in authoring mode or is projected on the table in execution mode.

The Authoring component contains all the classes that are needed for the authoring environment. These include the scene viewer, the annotation tools and annotations toolbar, the properties view, the physical objects view. The authoring component also manages user's interactions and handles mouse events for drawing annotations on the scene and adding physical objects to the scene. The authoring component also contains the classes needed for saving and loading ISAR applications. The Execution component contains the UI for loading the projects and the classes needed for running and ISAR application.

Both authoring and execution component use the Scene Renderer component for rendering the scene. In Authoring mode the rendered scene is shown in the scene viewer. In execution mode the rendered scene is on the tabletop using Projector component. Scene renderer queries the Scene and Annotations component for the current scene and renders all the annotations of the scene either on the tabletop image obtained from camera service in authoring mode or on the scene image projected on the tabletop in execution mode. For every annotations on the scene, the scene renderer calls the draw method of the corresponding annotations tool, which draws the annotation based on its properties values. The scene renderer is also responsible for coordinate transformations between camera, projector, scene, and physical objects. For annotations that are attached to a physical object, the scene renderer transforms their coordinates with reference to the position and orientations of the physical object.

The Scene and Annotations component contains abstractions for scene, annotation, physical object, and the project. In authoring mode the application creator creates instances of these abstractions using Authoring environment and saves them into an ISAR application. In execution mode instances are created from the loaded ISAR application. For all the physical objects that are added to a scene, the object detection and tracking component continuously updates the current position and orientations of the objects if they are physically present on the tabletop. When the objects are physically removed from tabletop, the tracking component also updated the list of scene objects that are present or removed from the table. The interaction component also updated the scene object mode based on user's interactions. For example it toggle the visibility of annotations, or starts video and timer annotations, or triggers the selection event of action buttons, and so on.

The Object Detection and Tracking component contains the classes for detection and tracking of the physical objects as well as hand tracking and tracking of the selection stick. For object detection and tracking this component acts as the abstraction layer between authoring and execution environment and the actual object tracking packages. The actual tracking of the different physical objects are performed by the object detection and tracking plugins. The object detection and tracking service delegates the object detection requests to all existing plugins and waits for their results to become available without blocking the main application thread. When the object detection results are available form any of the object detection plugins, a callback is called that updated the list of present physical objects of the current scene with the updated pose information. This design has three advantages: the computations intensive operations, such as markerless pose estimation does not block the main thread; different object detection plugins can work in parallel and as soon as any result is available the scene is updated; and the unified plugin structure and access API for object detection allows for adding new object detection packages to ISAR making it extensible for new application domains.

Finally, the Camera and Projector components hide the hardware specifics of the camera and projector hardware and do the image processing operations such as resizing and color conversion.

4.3.3 Control Flow

In order to achieve higher performance and maintain the desired frame rate, the object tracking, hand tracking, and selection stick tracking services run in their own separate processes. Each process also has a corresponding observer thread that waits for the results and calls the callback function when the result is ready. The interaction service, scene renderer, scene viewer and projection service also run in separate threads. Threads communicate through thread-safe queues with the main application thread. The processes communicate over multiprocessing queues with underlying OS constructs for inter-process communication.

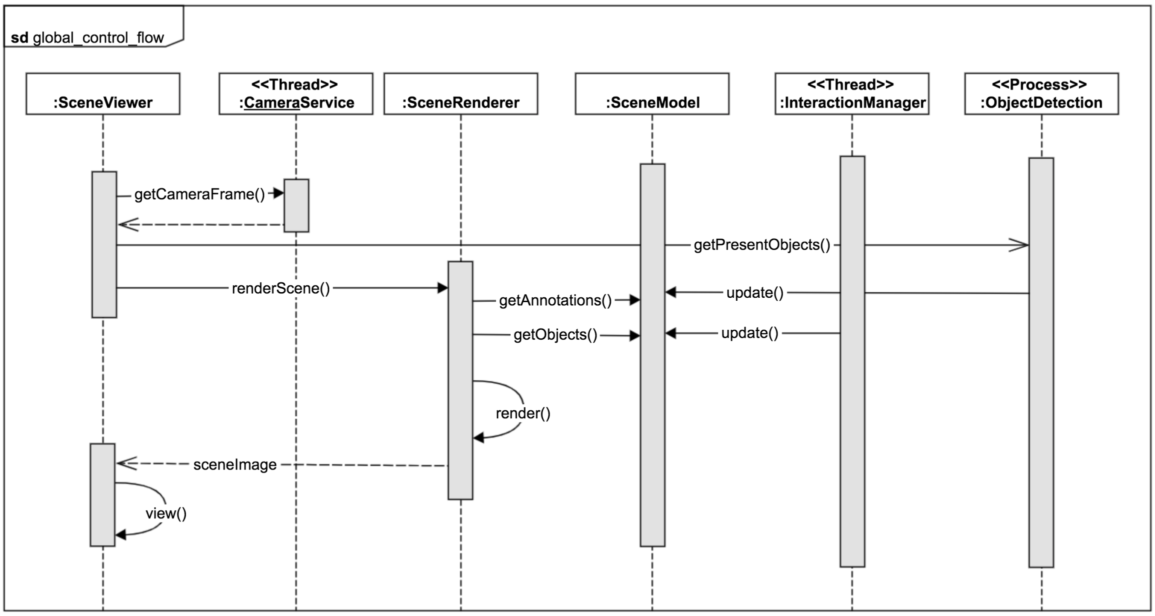


Fig. 4.18 The flow of control at each cycle of the tracking and rendering loop. The scene is updated by object tracking and interaction component and rendered by scene render.

Authoring and execution environments both run in continuous tracking and rendering loop that renders the scene for the scene viewer or for projection on the tabletop. The rendering and tracking loop must run at least 25 times per second to maintain the 25 frames per second requirement. Figure 4.18 shows the flow of control in each cycle of the tracking and rendering loop. The diagram shows the flow of control for Authoring mode. The Execution mode has the same flow of control at each cycle of tracking and rendering loop, with the exception that the scene is not rendered on the camera image of the tabletop but on an empty image that is projected on the tabletop. At each cycle, SceneViewer obtains the current camera frame from CameraService and passes it to object detection and tracking component. Object detection and tracking component runs in its own process parallel to the authoring or execution application and waits for object detection requests. An object detection request contains the image in which the pose of the objects must be estimated and address of a callback method that is called as soon as the results are available. The result of object detection updates the SceneModel. The scene model is the central repository which is updated by object detection and tracking and by interaction components. The updates include the new pose of object, and new state of the annotations such as their visibility, that is changed base on the defined interaction rules. The call to object detection and tracking is asynchronous. The scene viewer then sends a rendering call to the SceneRenderer. The scene renderer queries the SceneModel for the current annotations and physical objects and

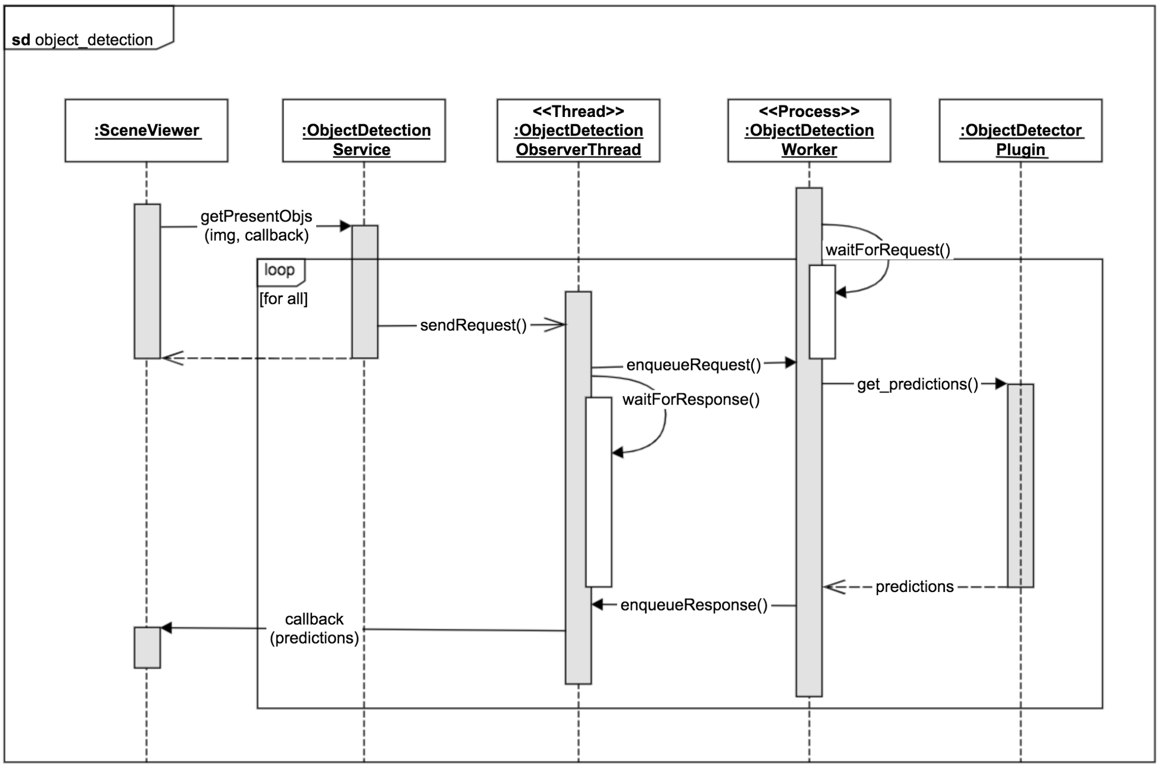


Fig. 4.19 The flow of control at each cycle of the tracking and rendering loop. The scene is updated by object tracking and interaction component and rendered by scene render.

renders the scene based on pose of the objects and state of the annotations. The result of the SceneRenderer is the rendered scene that is sent back to the SceneViewer to be shown in the Authoring environment or projected on the tabletop.

Figure 4.19 shows the details of the control flow for object detection and tracking. The object detection component is the facade [Gam95] that delegates the object detection request to object detection and tracking plugins. The actual object detection and tracking is performed by the plugins. For each object detection plugins the object detection service starts a worker process that loads the plugin. Each worker process maintains a request and a response queue. The request and response queues are multi-processing queues and use the underlying inter-process communication construct of the operating system. Each worker also has an observer thread that runs in the same process space as authoring or execution environment. The SceneViewer sends a non-blocking call for detecting the physical objects present on the tabletop to the ObjectDetectionService and passes the current image of the tabletop, and a callback to be called when the results are ready. The ObjectDetectionService delegates this call to all installed object detection plugins. For each observer-worker pair, the observer thread creates an object detection request containing the image of the tabletop and puts it into the request queue of the object detection worker and waits on the blocking response queue for the response to be available. The worker process that was blocked on the request queue picks the request from the queue and passes it object detection plugin for estimating the pose of the objects. The ObjectDetectorPlugin returns when the pose of the objects are estimated. The worker process packages the predictions in an ObjectDetectionResponse object and puts it into its response queue. As soon as the response is available in the queue, the observer thread continues the execution and calls the provided callback with the result of object detection. The callback then updated the SceneModel with the new pose of the physical objects.

4.3.4 Packaging and Distribution of an ISAR Application

An interactive tabletop application created with ISAR must be packaged and distributed to the application users to be executed in the execution environment. An application package is a folder on the file systems containing the following elements:

- **Project description file:**.The project description file (*project.json*) contains a dump of the application structure in JSON [Sev12] format. This file includes meta data about the project, such as its name, creation dated, and the authors name. It also includes a collection containing all the scenes, with their corresponding annotations and physical objects, event, actions, and interaction rules. Additionally, information about the projector and the camera, such as minimum required resolutions and the scene size are persisted into this file. The project description file contains all the information that the execution environment needs to load the project and project the scene on the tabletop.
- **Media files:** Video, audio, and image annotations are associated with a corresponding media file. These media files are automatically copied into the application package as the annotation is added to the scene.
- **Object detection and tracking plugins:** The physical objects added to the scene are provided by object detection and tracking plugins. For theses objects to be detected and tracked in the execution mode, that application package must contain the corresponding object detection and tracking plugins used to create the application. Including the object detection and tracking plugins into the application package, makes an ISAR application a self-contained package that can be distributed regardless of the configuration of the execution environment in which it should be executed.

At the start of an authoring project, the application creator is asked to select a folder on the file system where the application is saved. As the application creator continues with the authoring, the changes to the project description are automatically persisted. When application creator adds multi-media annotations to the scene, the corresponding media file (video, audio, image) is automatically copied to the application folder. When the authoring project is finished, the application creator selects the packaging command from the menu. The authoring environment then copies any object detection and tracking plugins, that were used by the application, into the application folder and compresses the folder into one single ZIP file, which application creator can distribute to the application users.

Chapter 5

Evaluation

ISAR provides end users with an authoring and execution environment to create interactive augmented reality applications for the tabletop. In this regard ISAR can also be considered as a toolkit confirming to the definition of a HCI toolkit by Ledo et al. [LHV^{+}18]: "A HCI toolkit is a generative platforms designed to create new interactive artifacts, provide easy access to complex algorithms, enable fast prototyping of software and hardware interfaces, and/or enable creative exploration of design spaces". Based on a review of existing HCI toolkits, Ledo et al. also identify four main evaluation strategies that are utilized for evaluation: demonstrations, usage studies, technical performance, and heuristics. Demonstration evaluations show what a toolkit can do and how it can be used to created a range of different applications that exhibit the purpose and design goals of the toolkit. Usage evaluations are used to show the usability of a toolkit often through user studies with a target group that performs a set of defined tasks. Technical performance evaluations validate how a toolkit addresses its non-functional requirements like accuracy and performance. Heuristics evaluations validate the toolkit against a given set of design heuristics such as Schneiderman's golden rules of interface design [SPC^{+}16] or Nielsen's heuristics [NM90].

In this chapter we report on the evaluation of ISAR based on a qualitative usage study with focus on usability and user experience of ISAR and two demonstrations cases. The purpose of the usage study is to show how users without technical background in development of interactive augmented reality applications can use ISAR to create interactive tabletop applications according to ISAR's use case model. The demonstration cases show how ISAR can be used to create interactive tabletop applications for different domains, such as manufacturing and medical rehabilitation.

5.1 Usage Study: An Interactive Tabletop Application for Vocabulary Learning

The purpose of this study is to show how ISAR can be used by non-technical end users to create interactive applications for the tabletop with significantly less effort and in significantly shorter time compared to development of such applications from scratch. Furthermore we want to show that such interactive augmented reality applications can be created by ISAR without any technical knowledge about low-level details of development of augmented reality applications. The result of this study also gives insights into how users would use ISAR and suggestions for improvements of the tabletop configuration and authoring and execution environment in the future versions.

The context of study is creation of an interactive tabletop application for vocabulary learning. We chose vocabulary learning as the target application for several reasons. The theory of multimedia learning [May02] suggests the combination of different modalities can improve learning. Multi-media annotations of ISAR can be used to provide the same information in different modalities. Furthermore, related work in the area of computer-supported language learning suggest that inclusion of context improves acquisition of vocabularies by supporting the associations between words and situations [STY$^+$14]. ISAR's support for combining physical objects and virtual multi-media annotations can be used to provide a rich context for vocabulary learning. Finally, educational applications are one of the main target application domains for ISAR and an interactive tabletop application for vocabulary learning demonstrates ISAR's support for educational applications.

5.1.1 Study Setup

In the context of creation of an interactive applications for vocabulary learning, we wanted to investigate the following questions:

1. Can a user without technical knowledge about augmented reality applications create an interactive tabletop application for vocabulary learning in a short (less that 1 hours) time? and without technical difficulty?

2. What challenges do the user's face in creating the interactive tabletop application using current version of ISAR? What are the improvement potentials of ISAR?

3. What is user's subjective impression about ISAR and how it facilitates the development of interactive tabletop applications?

We recruited 8 participants for creation of an interactive tabletop application for learning vocabulary using ISAR. The application should support learning of the nouns and verbs related to 10 tools from a simple household toolbox (examples can be seen in figure 5.1). We structured each session into four parts. First we interviewed the participants about their experience in software development and in particular experience with the development of the augmented reality applications. Then we did a short demonstration of ISAR features. Then we gave the participants a minimum specification for the interactive vocabulary learning application and gave them at most 1 hour time to create and test their application. Finally, we interviewed the participants about their insights and impressions of ISAR. The structure and questions of the study can be found in appendix D.

In order to gain insight into how end users use ISAR to create an interactive application we let the them design the applications freely. Not giving the participant a strictly defined task also has the advantage that it is more probable that different participants use different features of ISAR, and hence our usage study covers a larger subset of ISAR features. However, in order to make sure the participants apply a minimum of ISAR features for creating their application, an application description with minimum requirements for the interactive vocabulary learning tabletop application was given to them. This application description makes sure that the participants have an interactive application that has at least two scene with annotations and physical objects and contains interaction rules to define physical interaction of the application user with the tabletop. The application description is provided in appendix D. We provided a markerless object recognition and tracking package for detection of the tools, so that the participants did not have to create their own object recognition and tracking package.

Before each participant started with building the application, we did a 15 minute long demonstration of ISAR and what it can do by building a simple "Hello World" application. This demo application showed all the relevant ISAR features that the participants had to or could use for their applications, such as how to create scenes; how to create project; how to add annotations to the scene and configure them; how to add physical objects to the scene; how to define the scene navigation; and how to define events and actions and interaction rules. Example events and actions and interaction rules were demonstrated, but not all of them. It was mentioned that the participants can ask questions if they don't know what each event or actions is, although the names of the events and actions should be self-explanatory. It was also mentioned that the participants can ask for help if they faced any problems in creating their application.

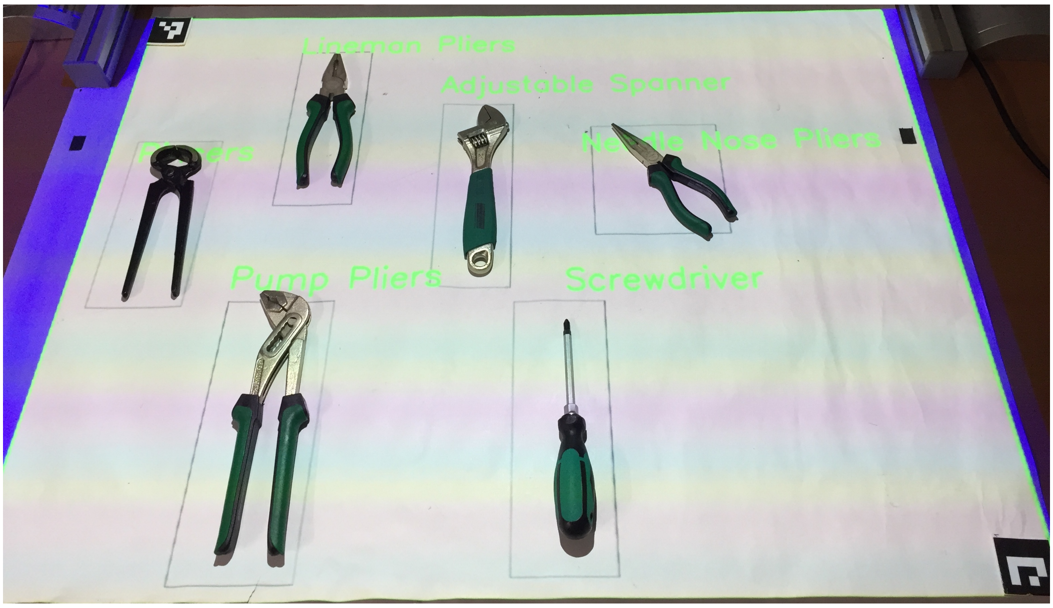


Fig. 5.1 Tools used for the vocabulary learning study.

One of the goals of this study is show that the relative time required for creating an interactive tabletop application using ISAR is significantly shorter compared to developing such application from scratch. We asked each participant about an estimate of time it would take if they wanted to develop their application from scratch without using ISAR. All participants of the study, except one, had more than two years experience in developing software, however none had experience in developing augmented reality applications. We then compared this estimate with the time it actually took for the participants to create the application using ISAR. This comparison is of course a subjective measure, nevertheless is gives us expert opinion about effectiveness of ISAR with regard to reducing the time and effort needed to develop interactive augmented reality applications for the tabletop.

To gain more insights about the usability of ISAR we also asked the participant to think aloud [Nie94] and explain their actions as they create the applications. The sessions were also recorded for further evaluation. The participants were also allowed to ask for help whenever they faced a problem, or needed further information about a feature of ISAR. After the participants were finished withe creation of the applications in ISAR Authoring environment, they tested their applications in the execution mode. They also had the chance to revise and adjust their application if they wanted to.

Each participant had altogether 1 hour to create and test his interactive tabletop application for vocabulary learning according to the given minimum specification (appendix D). After the participants were finished with creating and testing their application, we conducted the second part of the interview. The participants were asked questions regarding how they would estimate development of the application without using ISAR and about their experience regarding usability and usefulness of ISAR (appendix D). Finally they answered the Systems Usability Scale [Nie94] questionnaire to assess the usability of the system. The Systems Usability Scale questionnaire is an industry-standard questionnaire for quantitative assessment of usability of a software or technical system. The result of the questionnaire is a score between 0 to 100 calculated based on the weighted scoring of the answers. The details of the scoring is explained in appendix appendix D.

5.1.2 Results

Table 5.1 shows the demographics of the participants. Before demonstrating the functionality of ISAR, we asked the participants questions to indicate their level of technical affinity and familiarity with software development in general and with development of augmented reality systems in particular (Appendix D).

Participant	**Age**	**Technical Affinity**	**Software Development Experience**	**Experience with AR Systems**
P1	20-30	very high	expert	beginner developer
P2	20-30	very high	expert	only used
P3	30-40	very high	expert	beginner developer
P4	30-40	very high	expert	only used
P5	30-40	high	no experience	not familiar
P6	30-40	very high	expert	expert developer
P7	30-40	very high	expert	only used
P8	20-30	very high	expert	only used

Table 5.1 Demographics of the experiment participants.

All participants had age between 20-40 years with a high to very high level of technical affinity. The participants self-stated their level of technical affinity based on a question inspired by Affinity for Technology Interaction (ATI) Scale [WAF19]. All participants except P5 came from software development and research background, with more than two years of experience in developing software systems.

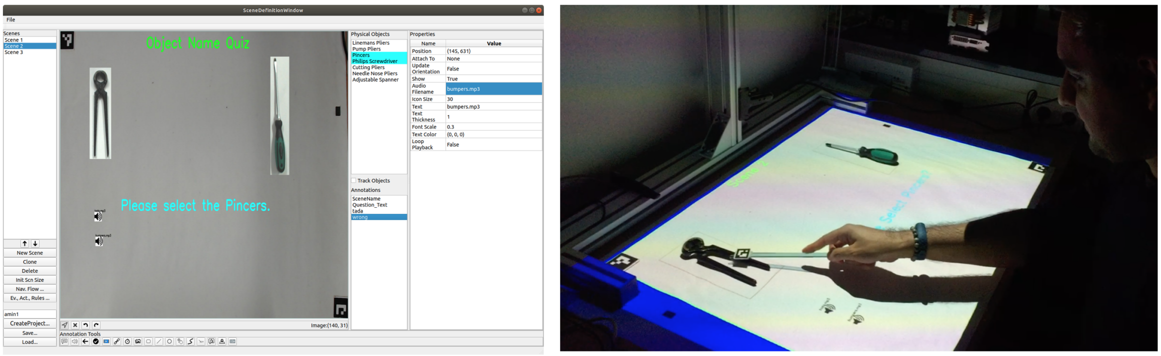
Two questions were asked to estimate the level of familiarity of the participants with augmented reality systems (Q1.4 and Q1.5). In the first question the participants stated their level of familiarity with AR systems with regard to using and development of such systems. Based on this questions one participant (P5) was not familiar with AR, four participants (P2, P4, P7, and P8) only used AR systems without knowing how they are created, two participants (P1 and P3) had created AR applications using frameworks like ARKit or Vuforia, and one participant (P6) was expert in developing AR systems including AR frameworks. The second question is an open-ended question about main components and main challenges for development of AR systems. With this question we intended to validate the answer to the question 1.4. The expected answer was an explanation of the augmented reality real-time loop including the tracking and registration (situated rendering) components. Our first expectation was that the participants who were not familiar with AR, or only had used an AR systems would not be able to answer this question. We also expected the participants with beginner and expert experience in developing AR systems (P1, P3, P6) would answer this question correctly. The first expectation was valid. Participants who were unfamiliar with AR, and those who only had used AR systems could not explain the main components and challenges of an AR system. To our surprise though, only P6 gave a confident correct answer to this question. Despite beginner development experience and using AR frameworks for creating AR applications, P1 and P3 could not confidently explain what are the components of an AR system. This may indicate that although AR frameworks facilitate the development of AR applications, merely using them in small projects is not a indicator of the competence in developing AR applications for end user requirements.

After the demographic questions in the first part of the study, we demonstrated ISAR's features to the participants using a small application. This demonstration took on average about 15 minutes and explained the main concepts of the ISAR such as authoring and execution modes, scenes, annotations, physical objects, and interaction rules. For this application we added a lineman pliers on the scene, attached an arrow annotation to it that showed its name. We also added an audio annotation for pronouncing the name of the line pliers and an interaction rule for playing the audio annotation when ever the pliers is pointed at using the selection stick.

After this demonstration we asked participants to create their own application for teaching vocabulary according to a minimum list of requirements (see 5.1.1 and appendix D). Each participant had maximum one hour to create his application. The participants were asked to think aloud as they created the application and the sessions were recorded for post-analysis.

All participants created applications with two or three scenes adhering to the minimum requirements. The scenes contained between one to three physical objects. All participants used text and arrow annotations attached to the objects for displaying their names. They also all used audio annotations for saying the name of the objects. Three participants (P3, P4, P6) used video annotations related to the physical objects available on the scene. Regarding the interaction rules, participants created from one up to six rules for scenes depending on the complexity of their idea and number of objects and annotations on the scene. The most frequent event used for interaction rules was *SelectionEvent*, which was mostly used to define rules such as "When the object is selected, play the audio annotation related to it" and "If correct object is selected, show positive feedback image". The latter rule was used for the cases where the participants had designed a scene which asked the user to select an object from two or three objects. For example P6 created a scene containing two physical objects (pincers and screw driver) and a text annotation asking the user "Please show which tool is pincers?". He then created two rules: "if pincers is selected, play audio annotation for positive feedback", and "if screw driver is selected, play audio annotation for negative feedback". P1 and P8 also created similar scenes using an *ObjectPlacementAreaAnnotation*. They asked the user to put a specific target object int the ObjectPlacementArea. If the correct object was put into the area, a positive feedback was shown or a corresponding audio was played. The second most frequently used event was *PhysicalObjectAppearedEvent*, which was similarly used for playing audio annotations or showing and hiding text annotations related to the target object. Two participants also used timer annotation on a scene and used *TimerFinishedEvent* to show correct answer to a question (P2) or hide an arrow annotation showing the name of an object (P6).

A general limitation we observed was that most of the participants created applications that were similar to demo application. This is possibly because they created their applications right after the demo and hence they were biased to use the features they were familiar with. Although none of the participants used the whole one hour for creating the applications, they still did not try many of other possible annotations types, and events and actions. The short time for the demo application and the large number of concepts and features explained in only 15 minutes can relate to that. Although we mentioned different kinds of annotation, events, and actions, we only used arrow, text, and audio annotations in our demo application and did not demo how other annotations, events, and action could be used. The complexity of the UI in terms of different features and usability issues, such as lack of tooltips for annotation tools could also have a limiting effect on the participants, so that they preferred to mostly try those features they had seen in the demo. P2, P3 and P6 however took an interesting approach. They first created a very simple application with one scene with exactly the same functionality as the demo application. After becoming confident with the UI they then created a more complex application with more than two scenes and multiple interaction rule in each scene.



Name	Value
Position	(145, 631)
Attach To	None
Update Orientation	False
Show	True
Audio Filename	bumpers.mp3
Icon Size	30
Text	bumpers.mp3
Text Thickness	1
Font Scale	0.3
Text Color	(0, 0, 0)
Loop Playback	False

Fig. 5.2 A scene designed by one of the participants in the authoring environment (left) and testing it in the execution environment (right). If the user selects pincers, a positive feedback sound is played back.

All participants first created their scenes and tested the functionality in the Authoring environment. They then used the execution environment to test their application as it is projected on the tabletop. On average they spent 30 to 45 minutes designing and testing their applications in the Authoring environment and 10 to 15 minutes running and testing them in the Execution environment. Figure 5.2 shows a scene designed by one of the participants in authoring and execution environment.

After the participants finished creating and testing their applications, we did a semi-structured interview based on seven questions to capture their general impression about usability and effectiveness of ISAR. We wanted to know which features of the ISAR participants liked most, and which difficulties and limitation they faced when building their applications. In terms of effectiveness, we also wanted to know the impressions of the participants on if ISAR can be used by end users to create interactive tabletop applications and if they would recommend ISAR. Table 5.2 shows the features of the ISAR the participants found most useful along with the limitations and difficulties they faces when creating their applications.

All 8 participants praised the WYSIWYG Authoring environment. They all liked the ability to define the scenes visually in exactly the same way as they would be projected on the tabletop. The participants also liked the ability to test the functionality of the scenes at the same time as they were designing. P2 said for example: "it is really grate that you can add annotations and objects and test right away how it would work for the user." The ability to add different kinds of annotations, specially multimedia annotations such as audio, was also praised by the participants. Four participants (P4, P5, P7, P8) already mentioned how it could be useful to create engaging stories and entertaining content for children. The participants also liked the ability to configure the properties of annotations from UI. P2 said for example: "It is like PowerPoint. You can use different shapes, audio, and video to create interactive

Useful Features	Limitations
WYSIWYG Authoring	Usability
• defining scenes	• too many features in authoring environment
• configuring annotations	• missing tooltips and inline help
• designing and testing at the same time	• positioning image for physical objects
• multiple scenes	
Interaction Rules	Definition of Rules
• no need for programming	• complex UI for defining interaction rules
• combining events with actions	• loosing overview with too many rules
Object Recognition and Tracking	Robustness of Object Tracking
• adding annotations to the objects	• robustness of markerless tracking
• events for object interactions	• light trade-off for tracking and projection

Table 5.2 The most useful features of ISAR as well as its limitations according to the participants.

slides that are projected on the table.", or P7: "I like it that I can configure everything in the visual environment without programming."

All participants liked the possibility to define interactions without programming and praised it with excitements. P4 said for example: "I think this is the best feature of ISAR with great potential.", or P3: "I really like this fast definition of rules without need for coding. For me this is the main part of the story [ISAR]." In general participants liked two aspects of defining interactions rules: the fact that they can define user's interactions without need for programming, and the fact that they can test their defined rules right in the Authoring environment. They also liked the ability to define more complex rules, for example the possibility to freely combine events and action and the ability to define composite actions.

All participants also found the markerless object recognition and tracking feature useful. They liked the fact that their application involves physical objects and physical interaction with the real world. They liked the ability to attach annotations to objects and used it for arrows, text, or even video annotations. They also praised the ability to define interactions based on object recognition and tracking events, such as *ObjectAppearedEvent* or *ObjectGroupAppearedEvent*. They found it specially useful that they could turn on object tracking in the authoring environment and test their defined scenes and interactions immediately.

Although all participants liked and praised main ISAR features (WYSIWYG authoring, interaction rules, and object recognition and tracking), however they also faced limitations and difficulties using these features. The main difficulties were rooted in usability issues. In the authoring environment the participants found that the UI is too complex and includes too many features in one single application windows (P1, P3, P4, P5, P7). They had sometimes difficulties to find the annotation tools, and mentioned that tooltips could help finding them. Also the all participants were missing the ability to move the template image of a physical object to change the layout of a scene. In the current version, if they user wants to move the template image on the scene, he has to turn on object tracking, and move the physical object on the tabletop, and then turn off the object tracking. By turning off object tracking the template image is shown at the lasted known tracked position of the physical object. We think however most of the usability issues in the authoring UI can be fixed with relatively minor effort, such as restructuring the layout of authoring UI or adding tooltips for different annotation tools.

The participants also faced difficulties for defining interaction rules. They found the UI for configuring events, actions, and rules complex and not user-friendly. For example they found the need for always selecting the scene for which an event, action, or rule is being defined, cumbersome. They also found it difficult to think ahead about all possible scenarios for defining the rules. For example P2 had defined a rule for a timer annotation, so that when the timer finishes a start button should be shown. However he forgot to add another rule for restarting the time whenever the scene is shown. Also, they found it difficult to keep an overview of all the rules when the number of rules in a scene was more that five. This can be possibly traced back to the long default names for the event, action, and rules. Most participants used the default names and did not change it to have a shorter, yet more descriptive name.

Participants also found robustness of markerless object recognition and tracking a slight source of limitations. Although the markerless tracking feature of ISAR is robust to occlusion and lighting condition under controlled light, still the tracking is sometimes not very accurate, specially for objects with few feature points. This was confusing for participants who expected tracking to work all the times. For example they wanted to have darker light conditions for a better projection of the scenes, which sometimes lead to weaker performance of markerless tracking. Nevertheless this issue can also be traced back to the unfamiliarity of the most of the participants with the challenges of markerless tracking. For example P6, who is an expert in tracking, praised the robustness of ISAR's tracking algorithm.

To have a more objective measure of the usability of current version of ISAR, we asked each participant to fill the Systems Usability Scale questionnaire [BKM08]. After evaluating the SUS value for each participant, we calculated the average SUS score. On average current

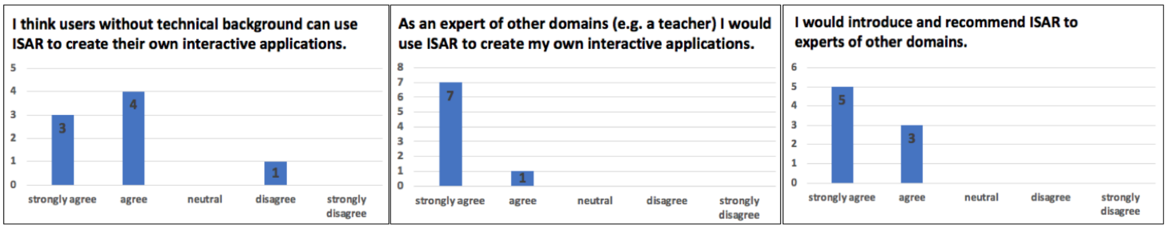


Fig. 5.3 Results of perceived impression of the participants about usefulness of ISAR.

version of ISAR achieved a score of 67 from 100, and receives a grade of C according to grading system by Lewis and Suro [LS17]. According to the categorization by Bangor et al. [BKM08] SUS score of 67 corresponds to "marginal acceptable" usability. This SUS score is an indicator of usability issues of ISAR Authoring environment and in alignment with our observations. One limitations of this evaluation must be noted though. We evaluated the SUS scores with the participants who are not the main intended target users of ISAR. The target users of ISAR are experts of domains other than computer science (for example a teacher) without technical background but with expertise on how ISAR could help them achieve their specific requirements. However, most of our participants came from computer science background and they work on creating technical systems on a daily basis. So we expect that SUS score of the current version of ISAR, if validated with end users without technical background would be less. Also it should be noted that SUS is most of the times used to evaluate the usability of production-ready systems [BKM09] as opposed to prototype implementation of ISAR. This is an issue with usability evaluation for prototype systems [GB08] and we believe fixing some of the issues with a small effort before evaluating ISAR with target users, would have a strong positive effect on its usability score.

We also asked participants an open-end question regarding in which domains they would consider most suitable for using ISAR. All participants referred to education and pedagogical domain as the most suitable. They suggested possible scenarios such as teaching concepts and words to pre-schoolers and in elementary school children, or using the engaging interaction for education of children with special needs. They also mentioned educational scenarios in the industry such as training for the tools and workflows. Entertainments, games, and marketing were among other domains mentioned by the participants.

Figure 5.3 shows the results of three questions we asked the participants to asses their perceived impression about usefulness of ISAR. The questions are in five-point Lickert scale and ask opinion of the participants about if they find ISAR easy to use for an end user without technical background, if they would use ISAR as an expert of domains other than computer science (e.g. a teacher), and if they would introduce and recommend ISAR to the experts of other domains. The general impression of the participants about usefulness of ISAR was positive (majority in strongly agree and agree category) and they would recommend ISAR. They however mentioned that their answers depend on the domain and scenario for which they would use ISAR.

Finally, considering the fact that most (7 out of 8) of the participants were experts in software development, we also wanted to know their estimation on how long it would take for them to create exactly the same application as they created, if they did not used ISAR and had to implement everything from scratch. This was an open-ended question which also involved discussion about if an off-the-shelf library for markerless tracking would be used or not.

Table 5.3 summarizes the response to this question along with the degree of experience of the corresponding participant with augmented reality and the short description about the complexity of the application he or she created.

Participant	**Experience with AR Systems**	**Application Complexity**	**Estimated Development Time**
P1	beginner developer	3 scenes; 6 objects; 16 annotations; 11 rules	6 to 12 months
P2	only used	2 scenes; 2 objects; 9 annotations; 7 rules	2 weeks to 1 month
P3	beginner developer	3 scenes; 4 objects; 7 annotation; 9 rules	1 to 2 months
P4	only used	2 scenes; 4 objects; 7 annotations; 6 rules	2 weeks to 1 month
P6	expert developer	3 scenes; 5 objects; 9 annotations; 7 rules	2 to 6 months
P7	only used	2 scenes; 3 objects; 12 annotations; 7 rules	1 to 2 months
P8	only used	3 scenes; 5 objects; 14 annotations; 9 rules	6 to 12 months

Table 5.3 Development time that each participant estimated for creating his application without using ISAR.

Al the developed ISAR applications were almost similar in terms of complexity (all adhering to the given minimum requirements. having 2-3 scenes), and all the participants finished their application in a time less than one hour. Nevertheless their estimate for the time they would need to develop the same application without using ISAR varies a lot, and they were relatively unconfident regarding their estimate . This can be traced back to the unfamiliarity of the most participant with development of augmented reality applications and its associated challenges. Only P6, who is an expert developer of tracking and augmented reality systems, was confident about his estimate. Also the participants gave different estimations depending on if they would use a tracking library or implement the tracking themselves. Regardless of the high variance in the estimates, the interesting fact for our analysis is: the minimum estimate for the development time is 2 weeks to 1 months. Of course we do not agree with this estimate tracing it back to the inexperience of the participant (considering the estimate of P6 to be more realistic). But this minimum estimate, even though unrealistic, gives a strong evidence that ISAR significantly reduces the time needed to develop such interactive augmented reality applications for the tabletop. Using ISAR all participant could create such interactive application in less than one hour.

Discussion

The results of the user study provides evidence that ISAR has achieved its main requirement in providing users without technical knowledge about augmented with a system to create interactive tabletop application fast and easy. The comparison of the time the participants required for creating their applications (average 30 minutes) with their estimate on how long it would take to create the same application without ISAR (average 5 months) is a strong indicator that ISAR reduces the time needed to create interactive tabletop applications by the end users. Furthermore, considering the fact that seven participants could not explain or were not confident in explaining the technical challenges in developing such applications, and comparing it with the applications they created, is an indicator that ISAR can facilitate development of interactive tabletop applications by hiding the complexities related to tracking, registration, and real-time performance.

The general impression of the participants about usefulness of ISAR was positive. They were excited about their applications and how easy they could create interactive applications involving the physical object and tangible user interactions. This is also reflected in their positive response about recommending ISAR to experts of other application domains and imagining how it could be used in the target domains, such as education and entertainment.

The study showed however, that the current version of ISAR has usability issues as reflected in the scores of system usability scale questionnaire (average score 67 form 100). Some of these issues, such as moving the template image of physical object on the scene using mouse, are minor and can be resolved by small changes in the code. Some others are however inherent in the design of ISAR, for example difficulty to keep overview of the interaction rules when there are many complex interactions intended. Although all the participants mentioned the ability to define interaction rules as a positive feature of ISAR, they all also mentioned the complexity of thinking about many fine-grained interaction rules and keeping an overview when the number of rules increases. To resolve this issue, a visual programming environment can be implemented that allows defining rules in a more user-friendly graphical representation.

We also observed that providing authors with best-practice guidelines can improve their experience and efficiency in creating ISAR applications. For example, although it was mentioned that the participants can define the name of scenes and annotations, we observed that they mostly used the default names for scenes and annotations. This however, lead to difficulties in keeping an overview when defining the rules, because the default names are not descriptive enough, for example, for distinguishing two annotations of the same kind. So a best practice recommendation would be to always name scenes and annotations. Also, we observed the participants had difficulty in keeping overview of a complex scene when many annotations and objects are added to the scene. A best practice here could be to define two or more simpler scenes, with fewer annotations and interaction rules instead and defining interaction rules for moving from one scene to another.

Finally, it should be noted that a limitation and a thread to validity of this study is its target participants. ISAR is intended to be used by end users without technical background and from domains other than computer science. Seven of the participants of the study were however from computer science domain. Of course, the purpose of the study was to gain insights into how ISAR is used, its perceived usefulness, and an estimate on how ISAR reduces the time and effort associated with development of tabletop interactive applications. In this regard, the participants could give us good insights into application usage and specifically their experience in software development gave us a more reliable comparison on how ISAR improves the development of interactive applications for the tabletop. Nevertheless a more formal study with a sample group of intended target users of ISAR (e.g. teachers of technical schools) is needed to reliably show ISAR has achieved its intended goals. Another limitation of this study is the coverage of different ISAR features. Although the participants used core ISAR features (scenes, annotations, physical objects, interaction rules) in their applications, however their applications involved a limited set of ISAR annotation types and interaction rules. Accordingly our claims about usefulness and difficulties of ISAR are only based on the limited set of observed applications and does cover the space of all possible applications that could have been created using ISAR. It should be noted though, that both kinds of the mentioned limitations are inherent in the evaluation of user interface toolkits [GB08, LHV$^+$18, OJ07].

5.2 Demonstration Case Studies

Demonstrations are the most widely utilized way of evaluating HCI Toolkits [LHV+18]. Compared to the research on artifacts that perform one single well-defined task, such as new algorithms or new interaction techniques, evaluating research on development of new toolkits is challenging. Toolkits provide their users with a large open-ended space of possible new applications that can be created by utilizing the toolkit [VH01]. This flexibility in creating new solutions, however, poses the generally inherent challenge for evaluating the effectiveness of the toolkit and showing its contributions and success [OJ07]. Demonstrations are a widely used evaluation strategy for HCI toolkits with the aim of exploring the open-ended space of possible toolkit applications and showing which applications can be crated by the toolkit and how the toolkit is used to create applications.

Demonstrations use selected scenarios form the space of possible applications supported by the toolkit and show how the toolkit can be utilized to fulfill that scenario. In this regard, demonstrations are a kind of proof-of-concept evaluation in support of the claim that toolkit's building blocks and design concepts enable users in creating the intended applications using the toolkit. Demonstrations can be used in order to show the breadth of possible applications that can be created using the toolkit (ceiling), or how easy it is for the users to start utilizing the toolkit to create their own applications (threshold), or simply show a step-by-step walk-through on how to use the toolkit [LHV+18].

In order to demonstrate how ISAR can be used to create interactive tabletop applications for domains other than education, we describe in this section two interactive applications created using ISAR. The first application demonstrates how ISAR can be used to create step-by-step workflow guidance applications [WON16]. A workflow guidance application guides the user through the steps of workflow to achieve a goal. We use the example scenario of assembling a computer mainboard for this demonstration.

The second demonstration shows how ISAR can be used to create applications that support medical rehabilitation, for example after a brain stroke. More specifically, we demonstrate how ISAR is used to create two applications that support hand-eye coordination exercises [HKD+13].

5.2.1 Workflow Guidance: Mainboard Assembly

We define a workflow as an ordered sequence of steps that need to be performed one after another in order to achieve a task goal. Each step of the workflow involves manipulating physical objects, such as materials and parts, and using tools. Using tools and manipulating objects requires learning manual skills that can be obtained by observation and training [PD95]. The workflows supported by ISAR are simple workflows with limited number of steps with clear step boundaries, and simple manual skills such as picking and placing objects and using tools. Examples of these simple workflows are manual industrial assembly. Complex workflows with unclear step boundaries and complex dexterities, like creating a wooden sculpture, are out of scope of ISAR.

Paper-based workflow instructions is the most widely used from of guiding through a workflow [FBÅM+14]. The instructions usually contain important information for preparation of the task, such as required tools and materials. Each step usually has at least a text description, that is most of the time accompanied with static images or images that show direction of movements using arrows. The textual description and images also contain important information that need be taken care of, for example orientations of parts when assembling a device. Video instructions are also often used for guiding through and teaching a workflow [Chi16]. Compared to paper-based instructions, video instructions are richer in terms of descriptions of the steps and showing how they are performed in action. The steps of the workflow are also described in the video, however usually quickly jumping to a step and gaining a quick overview of steps is more difficult in video instructions, because of the continuous linear timeline of the video [CLL+13].

Guidance through a workflow, specially industrial workflows such as prats assembly or maintenance and repair procedures, is one of the most typical use cases demonstrated in AR research [DBLS18]. Compared to paper-based and video instructions, augmented reality instructions have the advantage of presenting the information spatially registered right at the place where it is needed. This situated presentation of information leads to less cognitive load [PTTVG03] when doing the task [FKS16], because the user does not have to continuously change the context between seeking the information and doing the task, as is the case in paper and video-based instructions. Additionally, augmented reality workflow guidance applications can be designed and implemented to be interactive and context-aware [KFA+14]. Instead of presenting static predefined information, an interactive augmented reality applications presents information dynamically based on user's interactions and context information collected from other sources like environment sensors [ZON13].

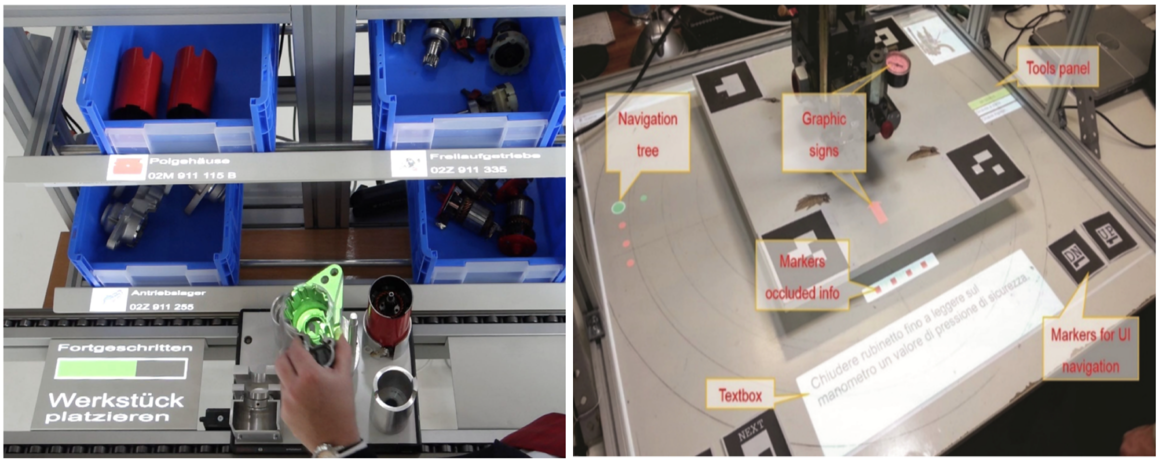


Fig. 5.4 Workflow guidance using camera-projector augmented reality setups. Funk et al. [FLM+18] (left) and Uva et al. [UFG+16] (right).

Different AR display technologies (stationary display, mobile AR, head-mounted displays, and spatial projection) have been used to guide workers through industrial workflows (2.1). Considering the requirement for bi-manual operation for most industrial operations, head-mounted displays (optical see-through and video see-through) and spatial projections are more often utilized because they allow bi-manual operation compared to mobile hand-held augmented reality systems [BMF+17]. We refer here briefly to some examples of spatial projections for workflow guidance.

Funk et al. [FLM+18] developed a stationary assembly guidance system consisting of a projector, a RGB camera, and a depth camera ((figure 5.4 a). At each workflow step the system highlights the corresponding material bins and tools and use the information from the depth camera to track workers actions, in order to automatically progress through the workflow. The system also features an authoring system based on Programming-by-Demonstration paradigm that automatically creates the steps of the workflow from an expert's demonstration. In a similar setup Ruther et al. [RHM+13] developed a system for guiding workers in the sterilization department of a hospital through workflows for sterilization of medical instruments. Their systems also features a camera-projector setup to project information related to workflow step on the workbench and a depth camera to track user's interactions with the projected user interface. Workflows are modeled as Business Process Models using BPMN 2.0 [All16] notation, and can be created using provided BPMN user interface. Uva et al. [UFG+16] developed an stationary camera-projector guidance system for assembly workflows (figure 5.4 b). The system uses RGB camera and marker-based tracking for highlighting different positions of the work piece at each step. Textual step descriptions are projected on the workbench and tangible cards with markers are used to navigate through the steps of the workflow. The workflows are represented in a tree structure with increasing detailed description and visual hints. However the system does not offer any authoring system for the end user. Camera-projector workflow guidance systems have also been implemented for tasks other than industrial assembly, such as cooking [UNT$^+$12, JHJL01]. These system also offer a similar set of feature such as projection of the workflow information on the working area, highlighting corresponding parts of the working area at each step, and tracking the user's interactions for interactive presentation of the information and progress through workflow steps.

The design of the workflow guidance support in ISAR is informed by the advantages, features, and shortcoming of these mentioned camera-projector workflow guidance systems. ISAR offers a rich WYSIWIG interface for end user to define the projection content for each workflow step as a scene, and define the interactive response of the system to user's interactions based on events, actions, and interaction rules. Adhering the simplicity design goal, ISAR does not require any specific hardware such as depth cameras and can be setup using off-the-shelf camera and projectors that are accessible in consumer market. Finally, ISAR integrates markerless object recognition and tracking.

In general for guidance through a workflow ISAR supports the following use cases:

- **Preparation:** It is typical for workflow instructions to have a first preparation step before the actual workflow steps [Chi16]. The preparation step usually describes what tools and material are needed to perform the workflow, and provide any other information that must be taken into account before starting the workflow. In ISAR the workflow author (the ISAR application creator) can provide these preparation information by creating a scene that contains the physical objects tools and materials. This scene can also have text, image, video, and audio annotations that instruct the application user how to prepare before the workflow begins. For each physical objects required for the workflow, ISAR projects the image of it on the tabletop. If the physical object is missing, the image shows an overlaid red cross, indicating the missing physical objects. The workflow author can also define an event for detecting when all the required physical objects are placed on the tabletop.

- **Description and information related to each step:** Each step of the workflow is associated with a scene. This scene can contain different annotations that provide the required information for performing the step. A text annotation can provide step description. An arrow annotation can indicate where to place objects or present important information spatially registered on physical objects. Video and image annotations can be added to show how to use tools or perform an action. Audio annotations can be used to provide step description or to provide audio feedback upon certain actions. At each step, the corresponding scene can contain interaction rules that define how the annotations behave upon user interactions.

- **Highlighting objects required in a step:** Research form camera-projector assembly guidance systems indicate that highlighting the materials, tools, and specific positions on the work piece reduce the cognitive load and increase speed of interaction during each assembly step [FSM$^+$15, BFSR16]. In ISAR, at each workflow step the application author can define rules for highlighting physical objects (tools and materials) that are required for the step as soon as the scene is shown. The workflow author can also add rectangle, circle, arrow, and other annotations registered to a specific position of the work piece to indicated where to place a part or perform an action. These annotations can be shown as soon as the step scene is shown, or become visible upon user actions such as picking a part or tool. By adding *ObjectPlacementArea* annotations, the workflow author can trigger events whenever an object is placed in a specific area on the tabletop or on the work piece.

- **Detection of errors:** Similar to guiding next action by highlighting parts and tools, providing feedback upon errors, such as picking wrong part, improves the speed of performing the workflow [FSM$^+$15]. Providing feedback on errors also improves the retention when learning the workflow in a training session [BFSR16]. ISAR supports detection and providing feedback when the application user is about pick the wrong objects. The workflow author can define an event that detects the user's hand is on top of or the user picked the wrong object. When this event is triggered the wrong object is highlighted red. Additionally, this event can be used to define a rule for playing back an audio feedback when the user is about to pick the wrong part or physical object.

- **Navigation through the steps:** ISAR offers different ways for navigating through the steps of the workflow. The workflow author can define the navigation flow by defining the sequence of corresponding scenes for every step. The navigation flow is used to define the next and previous scene for current step. By allowing specific definition of the navigation flow, the workflow author can define additional scenes that are not part of the default sequence of steps for the workflow. Examples of such scenes can be help or hint scenes. At each scene, the workflow author can add action buttons that navigate to next, previous, or back scene. When the user manually selects the action button, the target scene is shown. It is also possible to define actions for jumping to a specific scene, for example showing a help scene and then going back to the normal sequence of the workflow scene. Beside manual navigation through the steps of the workflow, it is also possible to define rules that trigger automatic navigation to next, previous, or any other scene. For example the workflow author can define a rule that indicates whenever the part is placed at a specific position (detected by an *ObjectPlacementArea* annotation), then show the next scene in the navigation flow.

The UML activity diagram in figure 5.5 shows the general interaction of the application user and responses of ISAR when performing the steps of a workflow. The application user first needs to load the ISAR project for the workflow. The first scene of the workflow project is the table preparation scene, containing all the physical objects required for the workflow. ISAR indicates the missing physical objects by showing a red cross on their image projected on the tabletop. The user then puts the required physical objects on the tabletop. When all the physical objects are put on the table, an action button indicating the start of the workflow becomes visible and the user can start the workflow. At each step of the workflow the corresponding scene is show. This scene includes all the information required for performing the step and the physical objects involved in the step are highlighted. The user performs the step by picking and placing the objects and using tools. The video annotations on the scene can show how to perform the step and use tools. Where to place an object on the work piece can also be highlighted. If the user picks the wrong object or tool, ISAR gives a visual feedback by highlighting the object red. The user follows the steps of the workflow by either manually navigating through the steps or the transition between steps happens automatically based on the interaction rules defined by the workflow author.

In this section we demonstrate how ISAR can be used to create interactive workflow guidance applications for the tabletop by using the example of a mainboard assembly workflow. The mainboard assembly workflow involves 6 parts and 1 tool: the mainboard, the RAM, the CPU, the heatsink, the CPU fan, the graphics card, and the screwdriver (figure 5.6). Before the workflow begins, the user must put all the parts and the screw driver on the table. The workflow consists of the following steps:

1. Place the CPU at the CPU socket on the mainboard. Pay attention to the correct orientation of the CPU; the CPU fits correctly into the socket only with the correct orientation. Lock the CPU by fixing the lock lever.

2. Place the heatsink on the CPU and fix it using screw driver.

3. Place the CPU fan on the heatsink and connect its power cable to the CPU fan power socket on the mainboard.

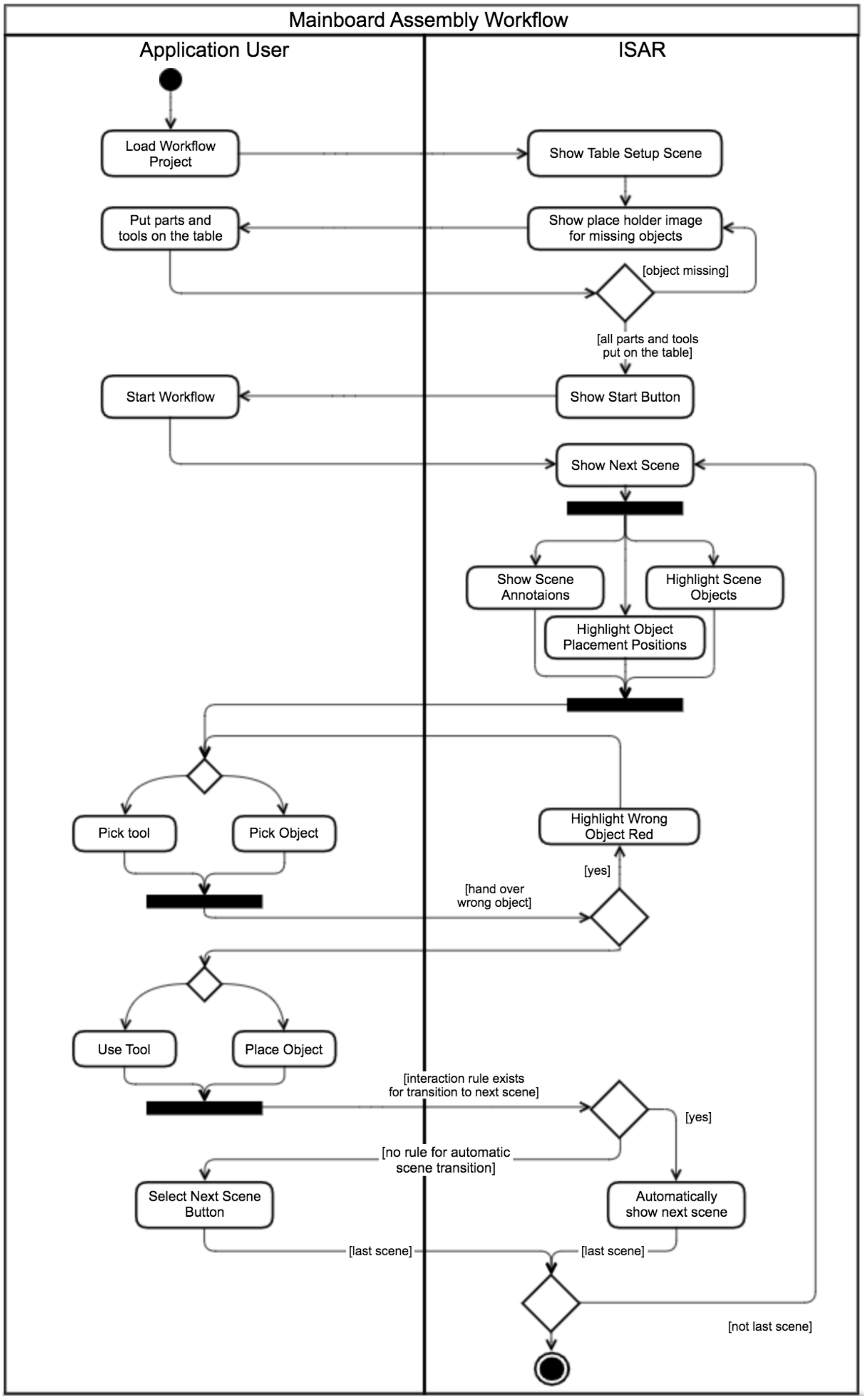
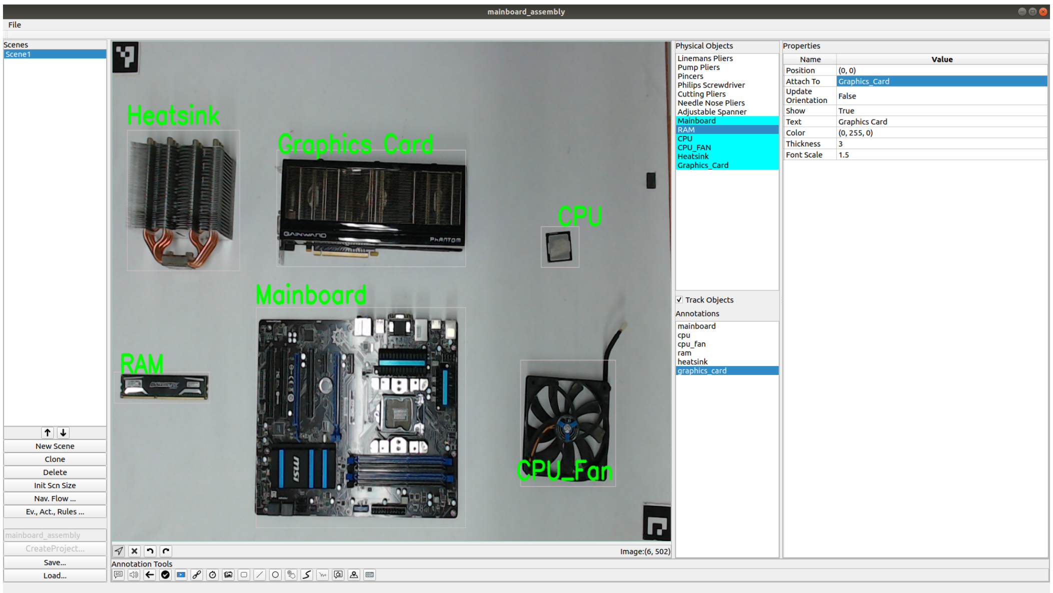

Mainboard Assembly Workflow
Application User
ISAR
Load Workflow Project
Show Table Setup Scene
Put parts and tools on the table
Show place holder image for missing objects
[object missing]
[all parts and tools put on the table]
Start Workflow
Show Start Button
Show Next Scene
Show Scene Annotaions
Highlight Scene Objects
Highlight Object Placement Positions
Pick tool
Pick Object
Highlight Wrong Object Red
[yes]
[hand over wrong object]
Use Tool
Place Object
[interaction rule exists for transition to next scene]
[yes]
[no rule for automatic scene transition]
Select Next Scene Button
Automatically show next scene
[last scene]
[last scene]
[not last scene]

Fig. 5.5 The mainboard assembly workflow. At each step ISAR projects the scene annotations and highlights the required objects. Transition to the next scene can either happen automatically base on user's interactions or by user selecting the next scene button on the tabletop.



Name	Value
Position	(0, 0)
Attach To	Graphics_Card
Update Orientation	False
Show	True
Text	Graphics Card
Color	(0, 255, 0)
Thickness	3
Font Scale	1.5

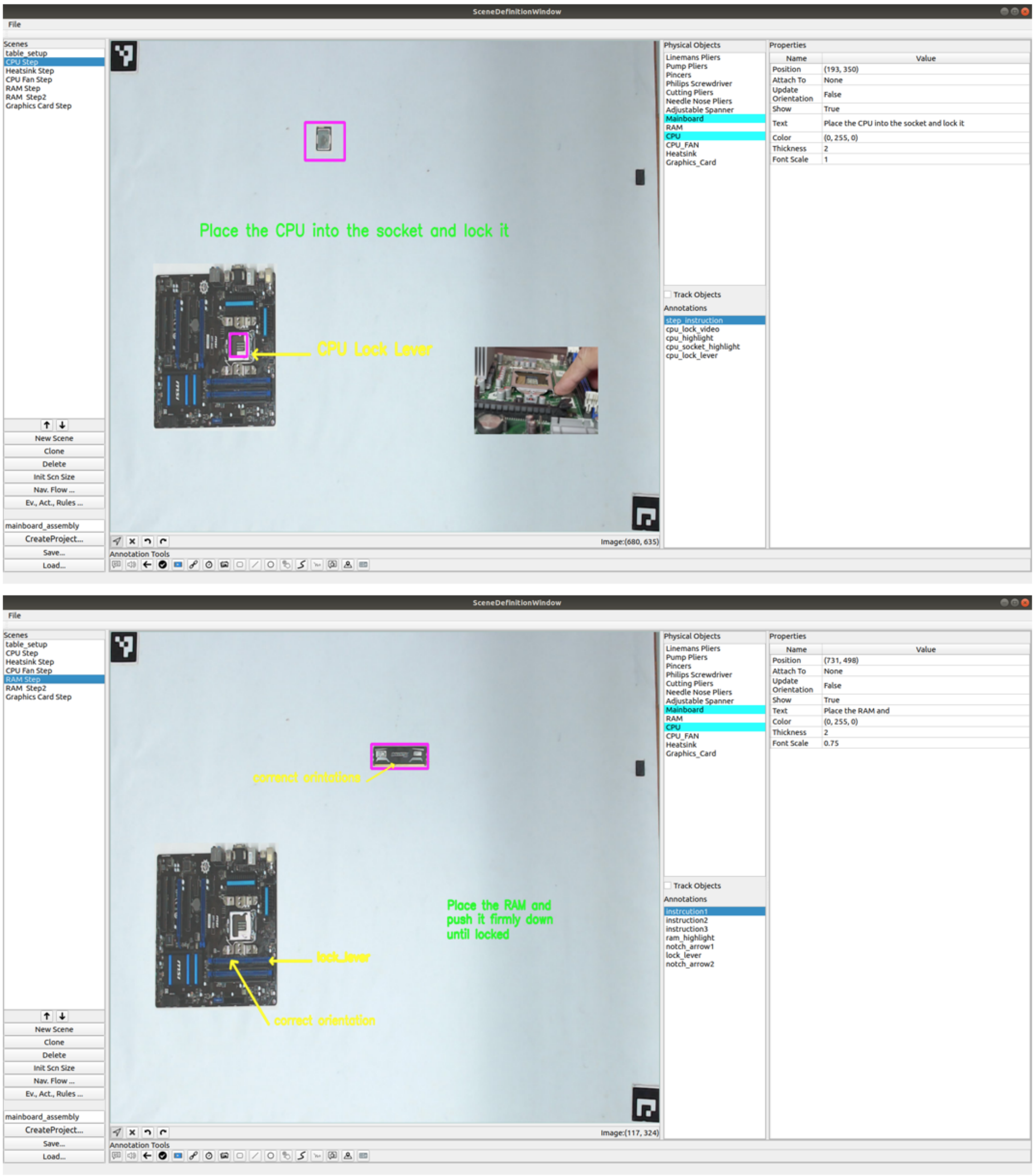
Fig. 5.6 The mainboard assembly workflow involves of 6 parts.

4. Place the ram module into the RAM bank on the mainboard. Pay attention to the correct orientation of the module, indicated by the notch; the RAM fits correctly only withe correct orientation. Lock the RAM by pressing it down and fixing the lever at each side.

5. Place the graphics card on the graphics car socket on the mainboard. Pay attention that the lock lever is fixed tightly.

The workflow author uses ISAR Authoring environment to create the ISAR project. For the workflow preparation a scene is defined that shows all the required parts and the screw driver. For each of step of the workflow a corresponding scene is defined that contains the textual description, arrows showing where to place the part or hinting on important information like the CPU lock lever, and a video showing how to perform the step. Figure 5.7 shows two scenes of the workflow. At the top, the scene corresponding to inserting the CPU is shown. The CPU is highlighted. The text annotation provides the step instruction. An arrow indicates the position of the CPU lock and the video annotation shows how to lock the CPU by fixing the CPU lock lever. At the bottom, the scene corresponding to the placement of the RAM is shown. Again the RAM and the RAM bank on the mainboard are highlighted. An arrow indicates the notch on the RAM module and gives hint for the correct orientation. An arrows indicates RAM lock lever and gives a hint for pressing firmly till the lever is locked.

The workflow author defines for each step scene a set of interaction rules that define how the scene changes upon user’s interactions. Table 5.4 shows excerpts of the interaction rules



Scenes
table_setup
CPU Step
Heatsink Step
CPU Fan Step
RAM Step
RAM Step2
Graphics Card Step

mainboard_assembly

Place the CPU into the socket and lock it

Physical Objects
Linemans Pliers
Pump Pliers
Pincers
Philips Screwdriver
Cutting Pliers
Needle Nose Pliers
Adjustable Spanner
Mainboard
RAM
CPU
CPU_FAN
Heatsink
Graphics_Card

□ Track Objects

Annotations
step_instruction
cpu_lock_video
cpu_highlight
cpu_socket_highlight
cpu_lock_lever

Properties

Name	Value
Position	(193, 350)
Attach To	None
Update Orientation	False
Show	True
Text	Place the CPU into the socket and lock it
Color	(0, 255, 0)
Thickness	2
Font Scale	1

Scenes
table_setup
CPU Step
Heatsink Step
CPU Fan Step
RAM Step
RAM Step2
Graphics Card Step

mainboard_assembly

Place the RAM and
push it firmly down
until locked

Physical Objects
Linemans Pliers
Pump Pliers
Pincers
Philips Screwdriver
Cutting Pliers
Needle Nose Pliers
Adjustable Spanner
Mainboard
RAM
CPU
CPU_FAN
Heatsink
Graphics_Card

□ Track Objects

Annotations
instrcution1
instruction2
instruction3
ram_highlight
notch_arrow1
lock_lever
notch_arrow2

Properties

Name	Value
Position	(731, 498)
Attach To	None
Update Orientation	False
Show	True
Text	Place the RAM and
Color	(0, 255, 0)
Thickness	2
Font Scale	0.75

Fig. 5.7 Defining the scenes for the steps of the mainboard assembly workflow. Top: Placing the CPU; Bottom: Placing the RAM.

Scene	Interaction Rules
Table Preparation	**E:** *ObjectGroupAppeared*(Mainboard, CPU, Headsink, CpuFan, RAM, GraphicsCard, ScrewDriver) **A:** *ShowAnnotation*(StartWorkflowButton)
	E: *ObjectGroupDisppeared*(Mainboard, CPU, Headsink, CpuFan, RAM, GraphicsCard) **A:** *HideAnnotation*(StartWorkflowButton) **A:** *ShowAnnotation*(SetupInstructionText)
	E: *SelectionEvent*(StartWorkflowButton) **A:** *ShowScene*(CPUStepScene)
CPU Step	**E:** *SceneEntered* **A:** *HighlightObject*(CPU, Green)
	E: *HandOverObject*(Headsink, CpuFan, RAM, GraphicsCard, ScrewDriver) **A:** *HighlightObject*(Headsink, CpuFan, RAM, GraphicsCard, ScrewDriver, Red)
	E: *ObjectPicked*(CPU) **A:** *ParallelCompositeAction*(*StartVideo*(PlacingCPUVideo), *ShowAnnotation* (CPULockLeverArrow), *ShowAnnotation* (RAMOreintationHint))
	E: *ObjectPlacement*(ObjPlcmntCPUSocket, CPU) **A:** *ShowAnnotation*(NextSceneButton)
RAM Step	**E:** *SceneEntered* **A:** *HighlightObject*(RAM, Green)
	E: *HandOverObject*(Headsink, CpuFan, GraphicsCard, ScrewDriver) **A:** *HighlightObject*(ActiveObject, Red)
	E: *ObjectPicked*(RAM) **A:** *ParallelCompositeAction*(*StartVideo*(PlacingRAMVideo), *ShowAnnotation* (RAMLockLeverArrow))
	E: *ObjectPlacement*(ObjPlcmntRAMBank, RAM) **A:** *ShowAnnotation*(RAMOreintationHint)

Table 5.4 Excerpts of the interaction rules defined in different scenes of the mainboard assembly workflow (not all the scene and the interaction rules are listed). **E**=Event **A**=Action

defined for three of the workflow scene. The interaction rules are used to show or hide annotations, start video, transit to the next scene, and so on. At the table preparation step, a rule is defined that checks when all the required parts and the screw driver are on the table. This is defined using an *ObjectGruopAppearedEvent*. When all the object are put on the table, the *ShowAnnotation* action is performed and makes the StartWorkflowButton visible. Accordingly, a rule is defined that checks if the an object is missing. This is checked using *ObjectGroupDisappearedEvent*, which is triggered as soon as any of the objects of the defined object group cannot be recognized on the table. If an object is missing the StartWorkflowButton is hidden, and the instruction text for the table preparation becomes visible instead. Another rule is also defined for when the user selects the StartWorkflowButton. Selecting the button shows the CPUStepScene, which is the scene corresponding to the first step of the workflow.

Interaction rules are defined similarly for other steps of the workflow. The first step of the workflow is the CPU steps. When the scene for this steps is shown, the CPU is highlighted with a green rectangle. If the user tries to pick any of the other parts, a visual feedback is projected by highlighting the wrong object red. This is detected using a *HandOverObjectEvent* that is checked for all the objects except the CPU. If the event is triggered, the *HighlightObject* action is performed for all the wrong objects. When the user picks the CPU, the video showing how to place the CPU in the CPU socket and how to lock it using the lock lever on the mainboard is shown. At the same time, the hint regarding the correct orientation of the CPU is shown and an arrow annotation becomes visible that show the position of the CPU lock lever on the mainboard with the short text indicating how to lock it. User's picking of the CPU is detected using an *ObjectPickedEvent*. Simultaneous starting of the video and making the hint and arrow annotations visible is done using a *ParalleCompositeAction* that gets the corresponding three actions for starting the video and showing the annotations as the arguments. When the CPU is placed in the CPU socket on the mainboard, the next scene buttons becomes visible. This is detected using an *ObjectPlacementEvent* that is triggered from an *ObjectPlacementArea* that the workflow author has added to the position of the CPU socket on the mainboard. The *ObjectPlacementEvent* has this annotation and the CPU as its arguments and is triggered when the CPU is placed in the object placement area. Table 5.4 shows only three scenes and for each scene also only some of the rules. For example the rules for transitioning back and forth between scenes are not completely shown in the table. Altogether for the mainboard assembly workflow we defined six main scenes and 28 interaction rules.

In execution mode the created workflow project is opened by the user and the user is guided through the workflow based on the scenes projected on the tabletop. Figure 5.8 shows the preparation scene (table preparation) and scenes corresponding to two steps of the workflow: placing the cpu step, and placing the RAM step. The annotations are projected on the table, including step description, videos, and arrows. The defined interaction rules at each step define how the annotations become visible and how the scene responds to user's interactions.

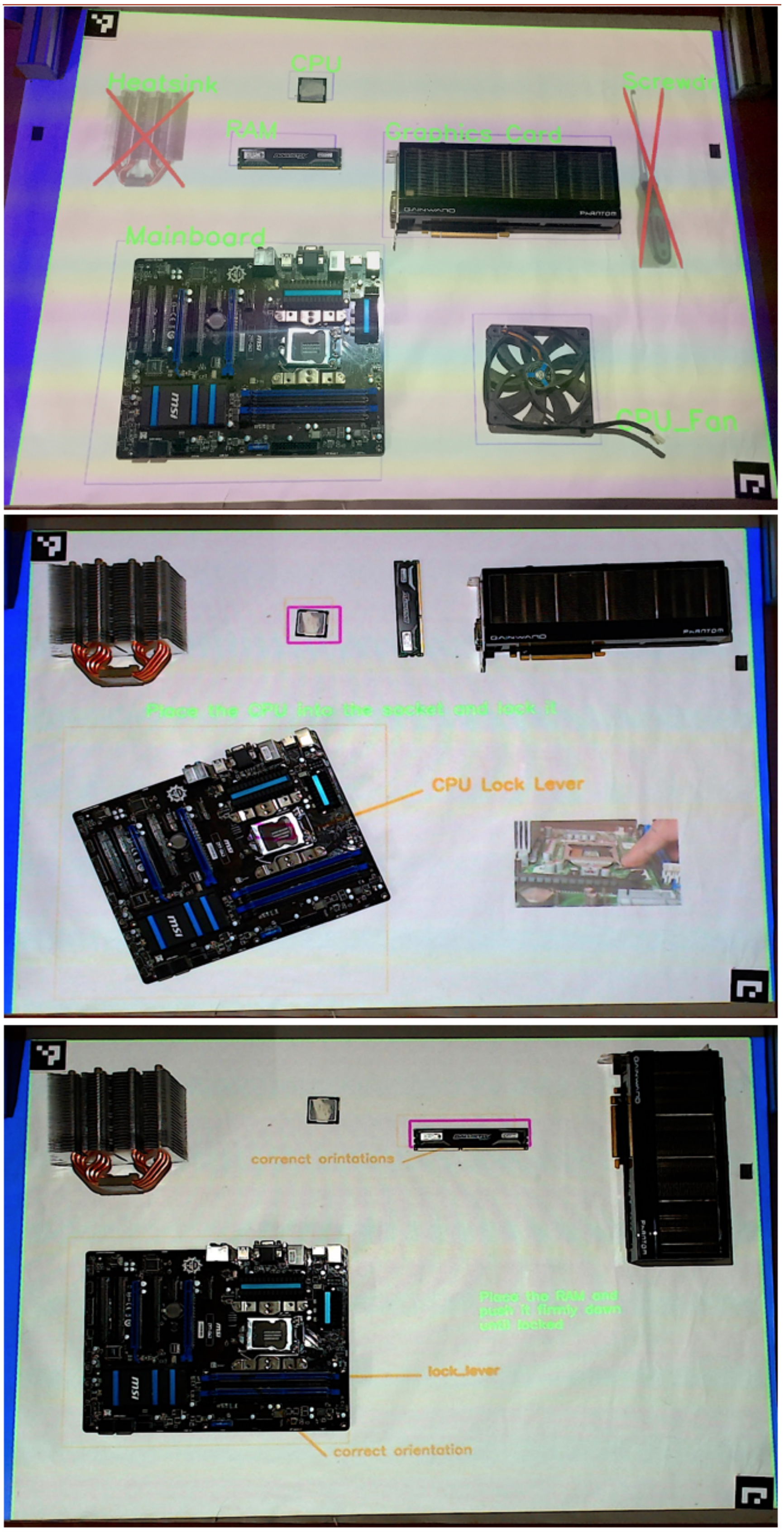


Fig. 5.8 The scenes associated with different steps of the workflow are projected on the tabletop during execution mode. Top: table preparation scene; middle: CPU step; bottom: RAM step.

5.2.2 Rehabilitation: Hand-Eye Coordination Exercise

By supporting physical interaction, combining digital content with physical objects, and the interactivity through interaction rules, ISAR can be used to create interactive applications for sensorimotor training and rehabilitation. Different augmented reality systems have been created to support motor tasks for physical or neuronal rehabilitation [AIRH12]. We refer here to some of the systems using spatial augmented reality and interactive tabletops.

Dunne et al. [DDLÓ$^+$10] developed three interactive games for rehabilitation of children with cerebral palsy on a multitouch table. The system features tangible interaction using a marker-based wand and a foam ball. The games support motor and cognitive training through moving and selecting objects on the screen. In a word games, the user should select letters moving around on the screen to spell the name of the animal shown. In another game the user must move a jar using the foam ball to move a jar under a butterfly and catch it by selecting the butterfly with the want as the jar is at the right place. The system does not however support authoring and creation of new content as well as adjustment of the difficulty and feedback by the therapists. Annett et al. [AAG$^+$09] also used a multi-touch tabletop for motor rehabilitation of upper extremity. Their system features different games, such as playing drums, popping up balloons, drawing, and selecting objects that appear randomly on the screen. The games are all based on touch interaction and no tangible objects are used. The therapists can adjust different features of the games, such as speed and difficulty, but there is no authoring system for creating new games by the therapists themselves. Mousavi Hondori et al. [HKD$^+$13] developed an interactive tabletop system for post-stroke rehabilitation. The system is based on a camera-projected setup and uses marker-based hand tracking. Their system supports performing of different tasks for the rehabilitation of upper extremity such as reaching, grasping, pointing, and tilting the wrist. They developed different games such as playing drums and changing shape of virtual objects projected on the table (figure 5.9). This systems also does not offer an authoring component for defining new exercises and creating new content. Augstein et al [ANRS$^+$13] used a multi-touch tabletop to develop a Tangram like game with tangible parts for post-stroke rehabilitation (figure 5.9). The system shows different patterns that must be created by the user by putting Tangram parts together. The system supports object recognition for the Tangram parts, and gives visual feedback for each piece that is placed correctly. Predefined templates for different difficulty levels, can be chosen by the therapist using the multi-touch UI. Similar to the system by Annett et al., this approach also allows different configurations of the game difficulty by the therapist, however, does not offer an authoring system to create new content.

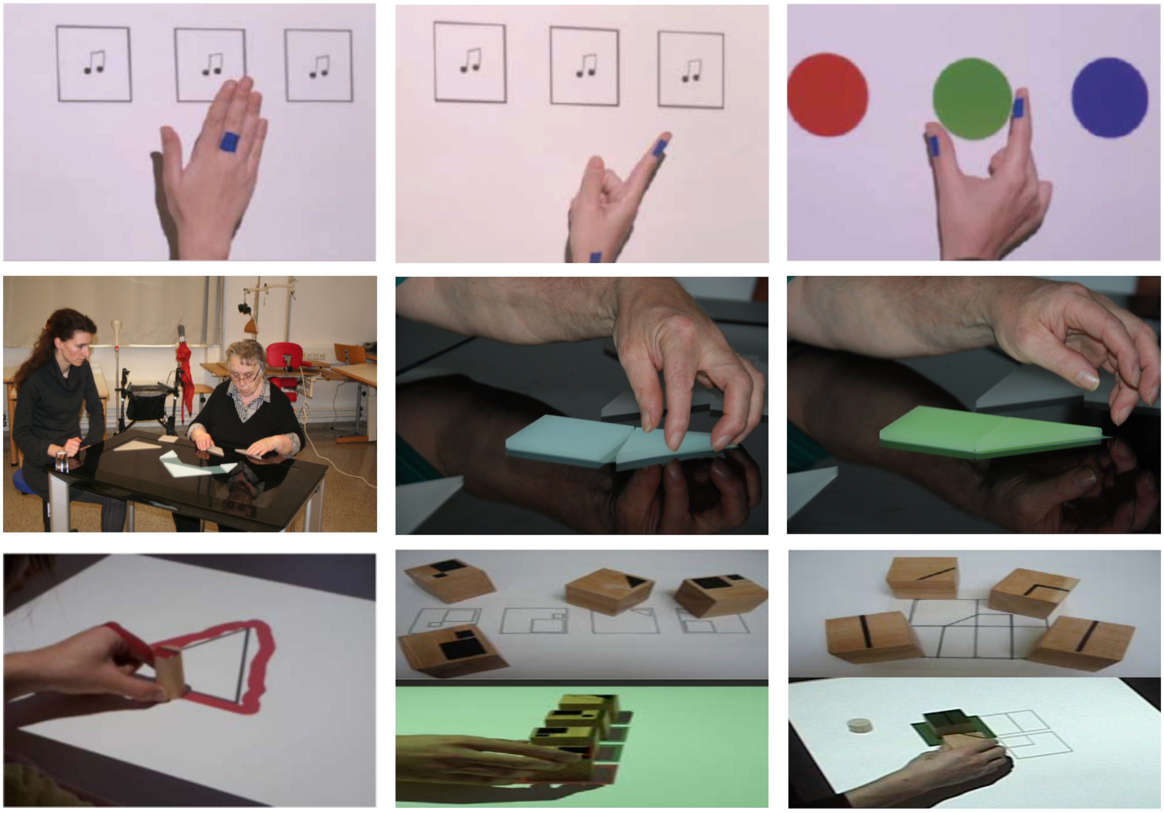


Fig. 5.9 Interactive tabletops for physical and cognitive rehabilitation of upper extremity. Exercises for reaching, pointing, and grasping by Mousavi Hondori et al. [HKD+13] (top); building a Tangram pattern on a multi-touch screen by Augstein et al. [ANRS+13]. (middle); following a projected pattern, sorting objects, and building a projected pattern by Leitner et al. [LTK+07] (bottom).

Tetteroo et al. [TVG+15] developed a tabletop system for cognitive and physical rehabilitation of upper extremity. Their system uses specific hardware called TagTile board. The TagTile board is a 2D grid of cells back-lit with LEDs. It uses object detection using RFID and can give audio and visual feedback. This system offers an end user development (authoring) system for designing the exercises. Therapist can define a timeline of actions for placing, lifting, and moving objects and the positions on the board where the objects should be placed. The audio and visual feedback of the board when the objects are placed can also be defined in the authoring system. Finally, Leitner et al. [LTK+07] describe three concepts for prototypes of a camera-projector interactive tabletop for post-stroke rehabilitation. The concepts were developed through focus group studies with therapist and support physical and cognitive rehabilitation by upper extremity exercises. The three concepts include copying a drawing projected on the table, sorting objects according to the projection on the table, and build a pattern by putting objects beside each other according to the projected instruction (figure

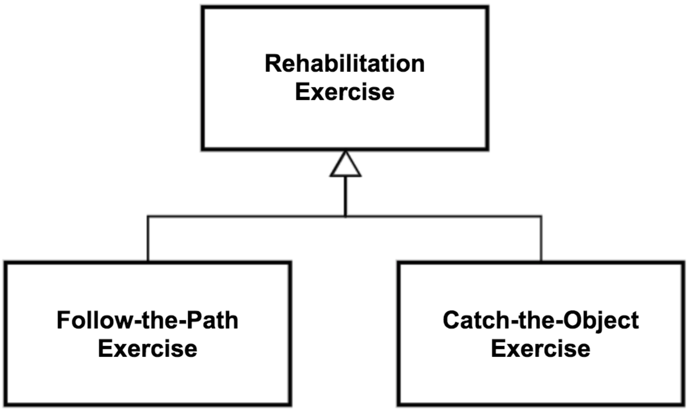


Fig. 5.10 Two kinds of rehabilitation exercises that can be created using ISAR.

5.9). Although the concepts for the three prototypes were developed through participation of the therapists, their approach does not offer an authoring system for the therapists to create new rehabilitation applications themselves.

ISAR can be used to create applications for physical and cognitive rehabilitation of the upper extremity similar to the above approaches. However, the main advantage of ISAR over the above approaches is that the physio-therapists are not dependent on the programmers for creating the applications and can use ISAR's authoring system to create interactive rehabilitation applications themselves. To the best of our knowledge, the approach by Tetteroo et al. [TVG$^+$15] is the only system that provides an end user development (authoring) system, that therapists can use to create their own interactive applications for rehabilitation. However this approaches requires a specific hardware (the TagTile board), and is limited in the way it allows the therapist to design the applications (the therapist can only define a timeline of simple actions such as grasping, moving, and putting of the objects).

Creation of rehabilitation applications for the upper extremity that involve reaching, picking, and moving and sorting objects is simple using ISAR's authoring system. For example to create a drums application, similar to Mousavi Hondori et al. [HKD$^+$13], the therapist can add different image annotations for drums to the scene and define rules that trigger different drum sounds when the user touches each image annotation with the hand. By integrating counter, timer, and feedback annotations in the scenes and defining corresponding interaction rules (e.g. counting how many time an action was performed, or defining scene transitions based on timer values, etc.) it is possible to create even more complex rehabilitation exercises.

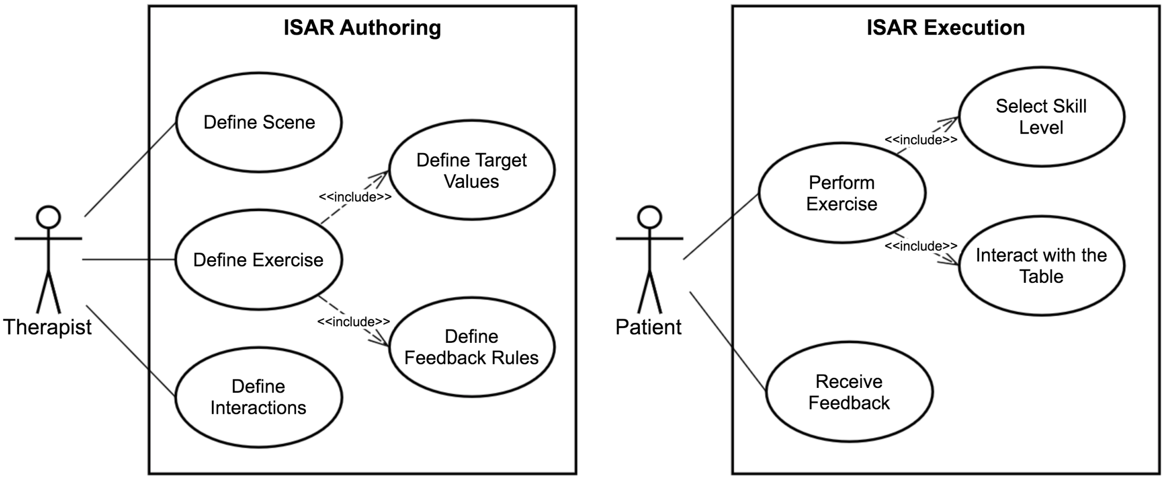
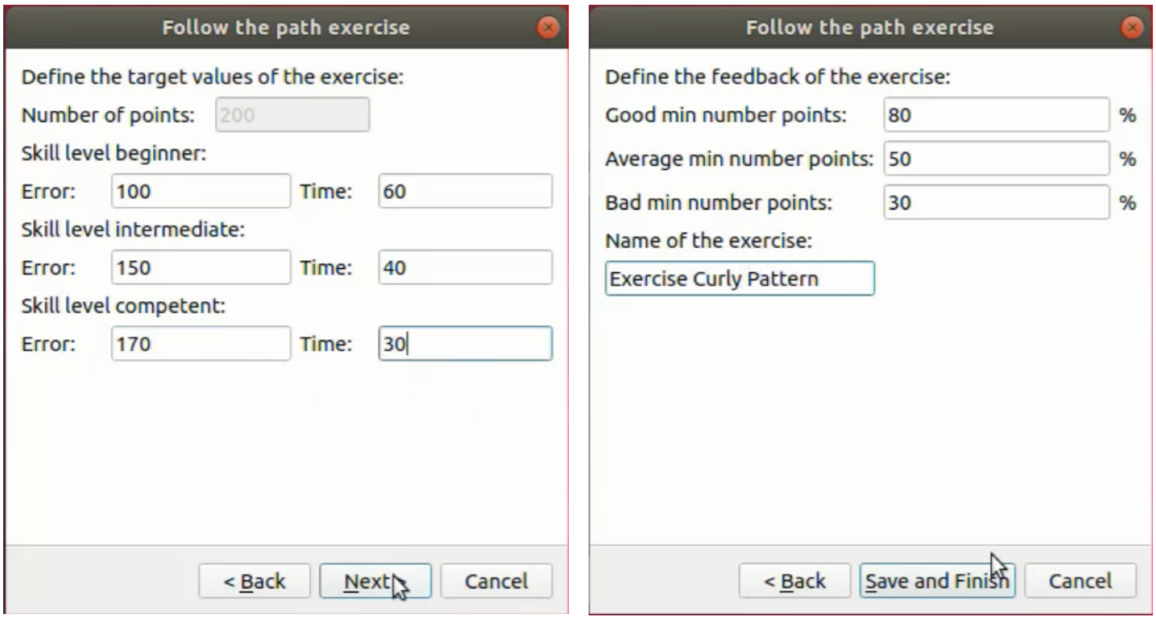


Fig. 5.11 A therapist uses ISAR's Authoring environment to create rehabilitation exercise. The patient uses ISAR Execution environment to do the exercises and receive feedback.

In the following, we describe how ISAR can be used to create two applications that feature hand-eye coordination exercises for the rehabilitation of the upper extremity (figure 5.10). Beside the possibility to define interaction rules, the therapist can define more fine grained feedback rules for these two exercises. In Follow-the-Path exercise, similar to the approach by Leitner et al. [LTK^{+}07], the patient must follow a curve pattern projected on the screen and receives feedback based on how accurate he could follow the path in the given time. In the Catch-the-Object exercise, similar to the approach by Annett et al. [AAG^{+}09], the user must hit images that are moving on the scene in an animation and receives feedback based on how many flies were hit in the given time.

In general, ISAR supports following use cases for creating and performing rehabilitation exercises (figure 5.11):

- **Defining Scene:** A rehabilitation exercise is an ISAR application consisting of one or more scenes. As with any ISAR application these scenes consist of annotations and physical objects and interaction rules and are defined in ISAR Authoring environment.
- **Define Exercise:** An exercise has an evaluation aspect that is used to calculate and give feedback to the patient. For example for the follow-the-path exercise the evaluation aspect is how accurate the patient follows the projected curve. Furthermore, ISAR supports different skill levels (beginner, intermediate, and competent). For each evaluation aspect, the therapist can define a target value that each skill level must achieve. The feedback is defined based on the skill-level and target value. There are three stages of feedback: good, average, bad. For each skill level, the therapist can



Follow the path exercise

Define the target values of the exercise:

Number of points: 200

Skill level beginner:

Error: 100 Time: 60

Skill level intermediate:

Error: 150 Time: 40

Skill level competent:

Error: 170 Time: 30

Follow the path exercise

Define the feedback of the exercise:

Good min number points: 80 %

Average min number points: 50 %

Bad min number points: 30 %

Name of the exercise:

Exercise Curly Pattern

Fig. 5.12 The therapist must define target values for each aspect of the exercise, as well as threshold values for different feedback levels.

define how many percent of the target values must be achieved for each feedback level. For each skill level, the therapist can also define the maximum time it is needed to achieve the target value of the exercise. The feedback is calculated when the exercise time is over. Figure 5.12 shows the dialog for defining target values and times for different skill levels and the feedback thresholds for the follow-the-path exercise.

- **Define Interactions:** The exercise scene can consists of different annotations and physical objects. Like other ISAR applications, interaction rules can be defined for how the users interacts with the content projected on the tabletop. For example for catch-the-object exercise, a rule can be defined that playback a sound whenever the user hits one of the objects. Beside defining target values and feedback rules, it is possible to define feedback using interaction rules. For example a rule can be defined that triggers a scene transition when the users has received good feedback for an exercise for a certain number of times. Such rules can be defined using events such as *CounterReachedTargetValue* or *FeedbackShown* and actions such as *IncrementCounterAction* and *ShowFeedbackAction*.

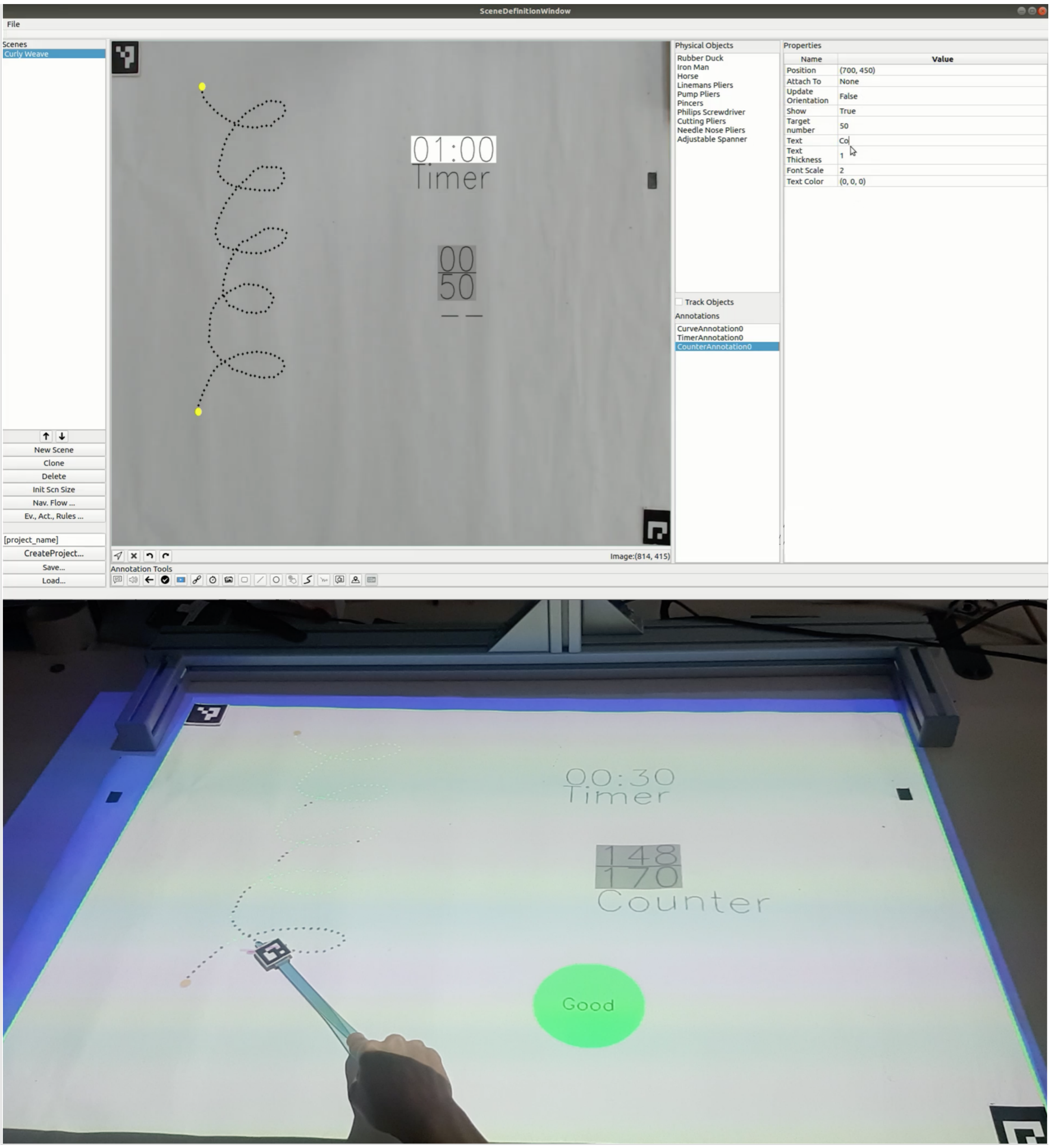
- **Perfom Exercise:** The therapist packages the defined rehabilitation exercise as an ISAR project. To perform the exercise the patient opens this project in the ISAR execution environment and selects the skill level he wants to be used for calculating the feedback. The scenes are projected on the tabletop and the patient can perform the exercise by interaction with the projected content. For example for follow-the-path exercise the patient uses the tracked stick to follow the curved pattern projected on the tabletop.

- **Receive Feedback:** When the defined time for performing the exercise is over, the feedback is calculated according to the patients skill level and the value he achieve for the defined evaluation aspect of the exercise. The feedback is then shown in a feedback annotation to the patient. Using interaction rule, it is also possible to give different real time feedback as the patient performs the exercise. For example, in follow-the-path exercise a rule can be defined that changes the color of curve points as they are being accurately traced by the patient. Or for the catch-the-object a rule can be defined that plays back a sound, as soon as each moving object is hit.

Figure 5.13 shows how a scene of the follow the path exercise is defined in authoring mode (top) and is projected on the tabletop in execution mode (bottom). In this exercise the patient should follow a curved path projected on the tabletop. For the follow-the-path exercise the application author (the therapist) must add at least a curve annotation, a counter, a timer, and a feedback annotation to the scene. For the curve annotation the author can define the total number of points that the patient should trace as he tries to follow the path. These point are then evenly distributed along the curve path. The maximum for the counter annotation is set to the total number of the points in the path. After the scene for the exercise is defined, the therapist defines the target values and time limit for each skill level (figure 5.12). For each feedback level (bad, average, good) he also defines a threshold for the percentage of the target value to be achieved. After defining the scene and the exercise the therapist saves the ISAR project.

When the patient loads the project, the exercise scene is projected on the tabletop (figure 5.13 bottom). The target value for the feedback and the value of the timer is set according to the skill level the patient selects when he loads the project. The patient follows the projected path using the tracked selection stick. The timer starts as soon as the patient traces the starting point of the curve. At each frame, ISAR checks if the stick passes over a point on the path. If a point is traced, its color turns to green to give real-time feedback to the patient, and the counter is incremented. When the timer is finished, the feedback is calculated based on the number of points the patient traced, the target value for his skill level, and the feedback threshold. The calculated feedback is shown in the feedback annotation.

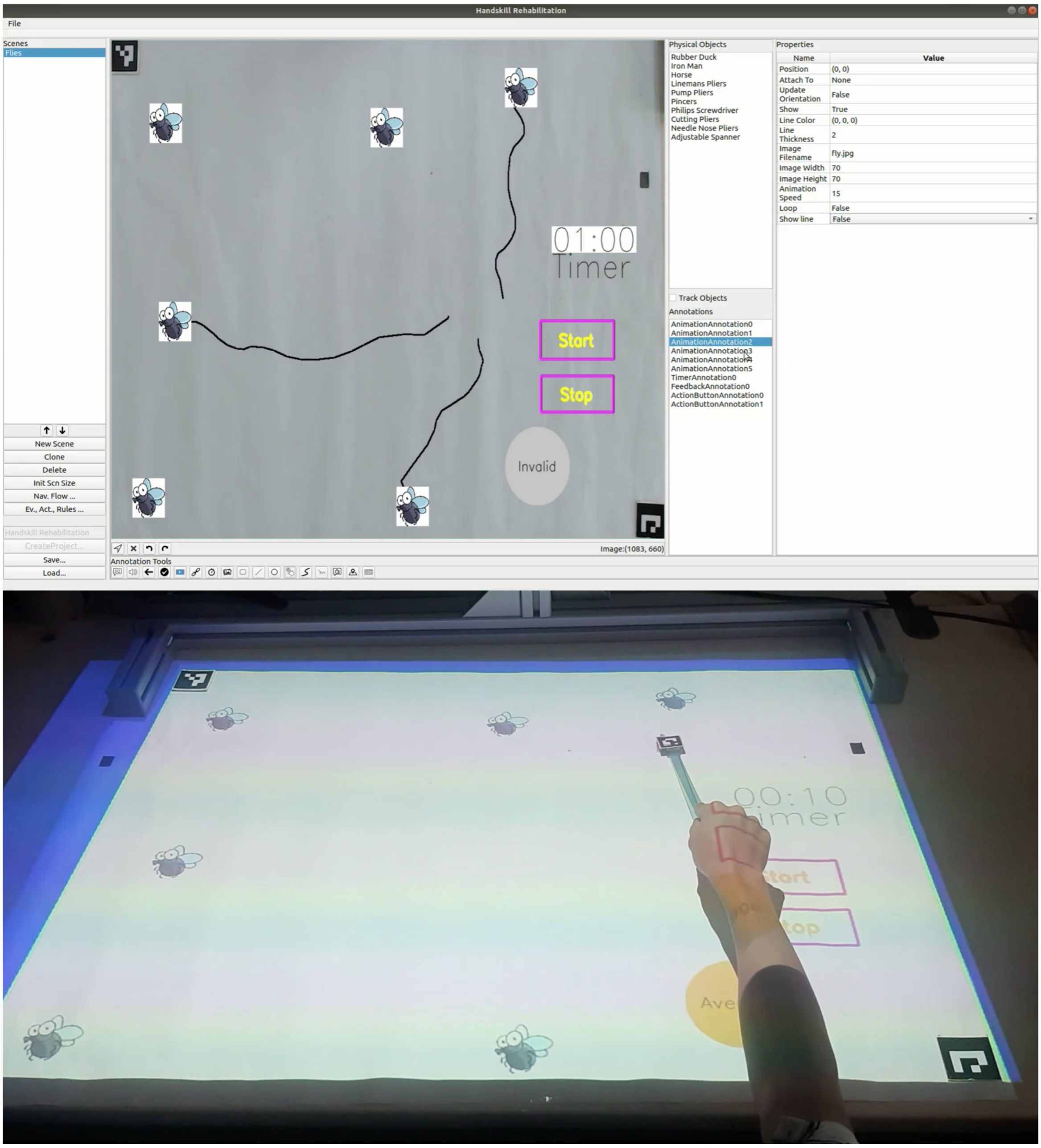
Figure 5.14 shows how a scene for catch-the-object exercise is defined ISAR's Authoring environment (top) and is projected on the tabletop in the execution mode (bottom). In this exercise the patient should hit moving images that are projected on the tabletop. The



Properties

Name	Value
Position	(700, 450)
Attach To	None
Update Orientation	False
Show	True
Target number	50
Text	Co
Text Thickness	1
Font Scale	2
Text Color	(0, 0, 0)

Fig. 5.13 Defining the scene for a follow-the-path exercise (top) and performing the exercise (bottom). To give the patient real-time feedback, the points of the curve turn green when they are correctly traced.



Physical Objects

- Rubber Duck
- Iron Man
- Horse
- Linemans Pliers
- Pump Pliers
- Pincers
- Philips Screwdriver
- Cutting Pliers
- Needle Nose Pliers
- Adjustable Spanner

□ Track Objects

Annotations

- AnimationAnnotation0
- AnimationAnnotation1
- AnimationAnnotation2
- AnimationAnnotation3
- AnimationAnnotation4
- AnimationAnnotation5
- TimerAnnotation0
- FeedbackAnnotation0
- ActionButtonAnnotation0
- ActionButtonAnnotation1

Properties

Name	Value
Position	(0, 0)
Attach To	None
Update Orientation	False
Show	True
Line Color	(0, 0, 0)
Line Thickness	2
Image Filename	fly.jpg
Image Width	70
Image Height	70
Animation Speed	15
Loop	False
Show line	False

Fig. 5.14 Defining the scene for a catch-the-object exercise (top) and performing the exercise (bottom). The patient must hit the flies as they move around.

feedback is calculated based on the number of objects the patient hits in the given time. To create a catch-the-object exercise the application author (e.g. the therapist) must add one or more animation annotations to the scene. An animation annotation moves an image along a predefined path. In figure 5.14 top, there are 6 animation objects, with a cartoon of a fly as the image. The author defines the path along which the image should move, using mouse in the authoring environment. This path can be shown or hidden. For each animation the author can define how fast the image should move, and if it loops from the beginning when reached the end of the path. Beside the animation annotations, the author should also add a counter, a timer, a feedback annotation, and possibly a button to start the exercise. Start of the exercise can also be triggered by any other interaction defined by the author in interaction rules. After the therapist designed the scene for the catch-the-object exercise, he can set the target value for number objects that each skill level should hit. For each skill level he also defines the exercise time and feedback thresholds (as in figure 5.12). He then defines any other interaction rules needed for the scene, for example when the start button is selected start the animations. After defining the scene and the exercise, the therapist saves the ISAR project.

To perform the exercise, the patient must load the exercise project into the ISAR execution environment and select his skill level. The exercise scene is projected on the tabletop (figure 5.14 bottom). When the patient starts the exercise, the animations start moving along the defined path the defined speed. The patient must now hit the animations as they move. The target value for the number of images to hit and how much time patient has, are set based on the patients skill level. When the time is over, the feedback is calculated based on the target value, how many images patient hit, and patients skill level. The calculated feedback is then shown in the feedback annotation.

Chapter 6

Conclusions and Future Work

6.1 Summary

In this dissertation we addressed the problem of authoring for interactive tangible tabletop applications by proposing ISAR: an authoring system for interactive tangible tabletops. ISAR allows end users, such as teachers and physiotherapists, to create their own interactive tabletop applications without the need to consult programmers. They can start exploring the proven potentials of augmented reality and interactive tabletop applications in their respective domains. We chose interactive tabletop platform for several reasons, including simplicity and familiarity of interaction on a physical table, advantages in terms of usability, ergonomics, and unobtrusiveness, and simplicity and low-cost and effort for the setup. ISAR is a tangible interactive tabletop setup and a corresponding authoring system for creating applications for it. The hardware configuration consists of a consumer projector and a webcam hanging on top of the table. The requirements and features of ISAR were extracted based on a review of related work, and insights gained from different related projects, that were implemented in the course of this dissertation. ISAR consists of two modes: authoring mode and execution mode. In the authoring mode, the application creators use a WYSIWYG authoring environment to design and create their interactive tabletop application as a sequence of scenes. The author can design each scene by adding physical objects and different multimedia annotations, such as text, image and video, as well as geometric shapes such as lines, squares, and arrows. Furthermore, the author can define the interactions of the user with the table and the physical objects and define the response of the scene in form of interaction rules. A rich set of events and actions allow defining complex interaction rules. ISAR supports both markerless and marker-based object recognition and tracking and offers a simple tool for creating tracking models for markerless tracking and associating markers with object for marker-based tracking.

ISAR was evaluated by a user study and two demonstration cases. In the user study we asked the participants - mostly experienced software developers - to create an interactive application for learning a vocabulary of technical terms. We did a usability evaluation and reported on positive features, as well as challenges the users faced in creating their application. The results showed usability issues of ISAR authoring environment, as well as its most positive features: WYSIWYG definition of scenes and definition of interaction rules. The participants reported it takes significantly longer to create a similar application without using ISAR authoring. In the demonstration cases we showed how ISAR can be used to create applications for workflow guidance and medical rehabilitation.

6.1.1 Limitations

Some limitations of the proposed framework and authoring tool are inherent results of the design decisions made. The protective augmented reality setup is a 2D interface and cannot present 3D content, which could possibly be more useful for use-cases such as assembly guidance. Also, the available area for augmentations is limited to the surfaces of a tabletop and hence limited to only the activities that can be performed on a tabletop. Furthermore, as simplicity and domain independence were among our design goals, the trade-off between threshold and ceiling of the framework can be a limitations: the users can setup the table very easily and learn to use the authoring application fast, but the interactive applications created with ISAR can be limited in their complexity and may not support specific features required for a domain (e.g. education).

ISAR has to be used in controlled lighting conditions. Although the tracking algorithms (markerless and marker-based) are relatively robust to different light intensities, nevertheless a trade-off must be made between the lighting requirements for projection versus object recognition and tracking. In a too bright environment the visibility of projections degrades, whereas in a too dark environment the tracking algorithms are no more robust. We tested ISAR in all scenarios in a typical office illumination environment with florescent ceiling lights, which offered good conditions for both projection and tracking.

Another limitations that was also observed in the user evaluation, was about creation of interaction rules. The user may be challenged in anticipating all possible interaction events that need be addressed. Of course the WYSIWYG authoring and testing environment can be helpful in addressing this limitation, since the user can directly test what he is creating. Also, the number of interaction rules may become too much and hard to handle with the current user interface. Additionally, although we have offered a rich set of events and actions that cover many possible scenarios, there might be the need for more complex events and actions. In this regard the extensibility of the framework can help in addressing new requirements.

Finally, the evaluation approach can be a limitation and thread to validity of our claims. ISAR is targeted at end users without technical background, but we did not evaluate it with this target group, but rather with experienced software developers. Although this limitation is inherently present in many other toolkit researches [LOG17, GB08], still ISAR must be evaluated in a longer study with the non-technical target groups.

6.2 Future Work

The adoption of interactive augmented reality and tabletop applications in everyday practice of the experts of different domains requires technologies that are accessible and simple enough to be used by non-technical users. Authoring systems for augmented reality, including the current work, are a step towards this direction. Nevertheless, no single work can have a large impact on its own and a collective effort of a community of researchers and technologists is required. Here we discuss a few future directions on how ISAR can be extended and enhanced towards this vision.

An interesting direction for further research could be implementation of a platform for distribution and sharing of ISAR applications. Such a platform allows for crowd-sourced creation of content and interactive augmented reality applications for the tabletop. The execution environment of ISAR in combination with the distribution platform could build up a system similar to the concept of AR Browsers [GLG11], however for the interactive tabletop. Leaning on the idea of AR 2.0 [SLB11], such a platform could create a community in which users create interactive application and can rate and comment on applications of others. Facilitating long-term research on the adoption and usage of interactive augmented reality by the end users in the wild would be another advantage of such sharing and distribution platform for ISAR applications.

Extending and enhancing ISAR with specific focus on educational scenarios can be another direction for future work. Educational settings are one of the main application domains that can benefit from augmented reality and interactive tabletops [DD14]. Interactive tabletops offer unique possibilities and have been used to support collaborative learning. ISAR can be extended to support collaborative learning scenarios in many different aspects. Currently, the interaction with the table is designed for a single user. To support collaborative learning, multi-user interaction and support for user identification can be added to ISAR. Furthermore, the execution environment can be equipped with interaction hardware and software sensors that can capture application user's interaction and provide this data to teachers for online or offline learning analytics. Another area for extending ISAR for educational settings is to add support for better definition of exercises and evaluation of learning progress. Currently, a teacher can define different kind of exercises, such as multiple choice or object placement exercises, by using annotations and interaction rules. An extension to support exercises can be providing a UI for definition of more complex exercise and evaluation types and provisioning of feedback that is based on pedagogical considerations, such as scaffolding. Additionally, ISAR applications can be integrated into a Learning Management System that allows students and teachers to monitor the progress of learning and provide new learning content accordingly.

ISAR could also be extended for new modalities for user interaction. Integration of a speech recognition component into the framework could allow for speech interaction of the application user with the table. This can be used for example to issue different actions, such as navigating scenes, or in the educational scenarios to define new kinds of exercises. Integration of multiple camera or a depth camera could also allow for 3D interaction on the tabletop and better hand and object tracking.

Another possible direction for future work could be adding the possibility for real-time remote collaboration to ISAR. Similar to remote collaboration systems such as TeleAdvisor [GLC15] and the remote collaboration tool by Guaglitz et al. [GNTH14], the authoring and execution environment of ISAR could be extended to allow a remote user, such as an expert or a teacher, to add annotations to the scene in real-time. The remote worker or student would then see the annotations in real-time as the scene is being projected on the tabletop. By registering the hand of remote expert on the scene projected for the worker, it can also be possible to guide the worker using hand interactions. Similar approaches have been implemented for example in [RT07], [KSF06] and [SMF16].

Appendix A

Annotations

Annotations in ISAR can be grouped into geometric shapes, multimedia annotations, dynamic annotations, and interaction annotations. In the following, we list the different annotations in each group and their corresponding properties, events, and actions.

All annotations are subclasses of the abstract superclass `Annotation`. The abstract class `Annotation` provides the following properties for all annotations: `Position`, determines the 2D position of the annotation in scene coordinates or object coordinates in case the annotation is attached to an object; `AttachTo`, the physical object that the annotation is attached to or `none`; `Show`, determines if the annotation is shown on the scene or is hidden; `UpdateOrientation`, in case the annotation is attached to a physical object, determines if the orientation of the annotation should be changed based on the pose of the object.

All annotations also support the following actions: `ToggleAnnotationVisibility`, `ShowAnnotation`, `HideAnnotation`. Many annotations also support the `SelectionEvent`, that is fired when the selection stick or the hand is over the annotation for the define time interval for triggering the `SelectionEvent`. The time interval for triggering the `SelectionEvent` is defined in ISAR's settings.

A.1 Geometric Shapes

A.1.1 Line

Draws a line on the scene with the given thickness and color. Figure A.1 (1).

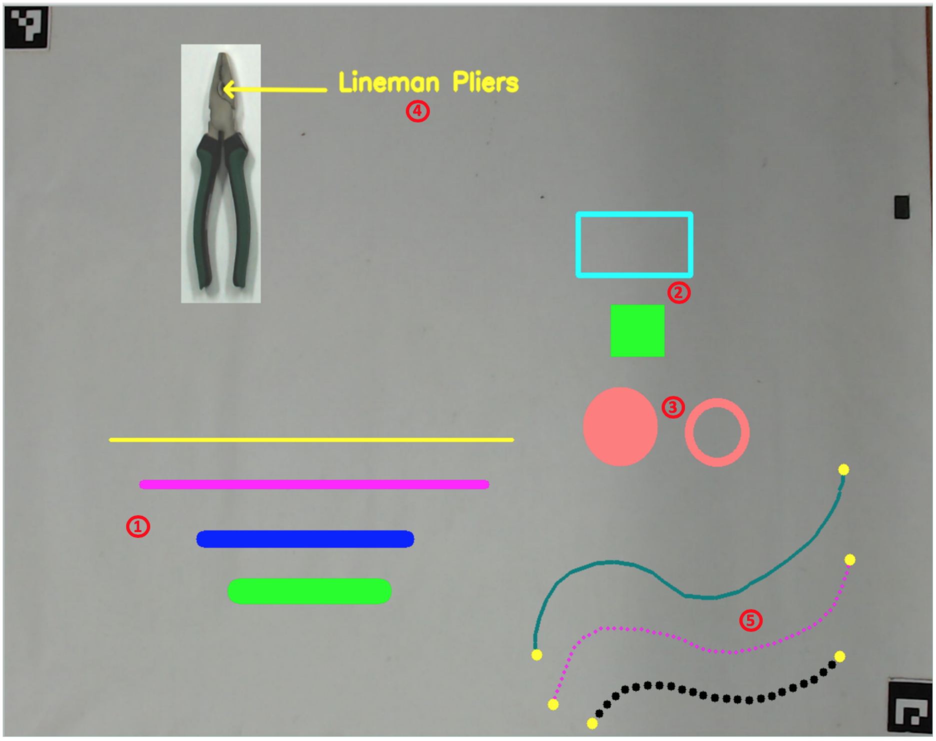


Fig. A.1 Geometric shape annotations.

Properties: `Start`, the 2D coordinates of the start point of the line (has the same value as `Position`) • `End`, the 2D coordinates of the end point of the line • `Thickness`, an integer value indicating the thickness of the line • `Color`, the RGB color of the line.

A.1.2 Rectangle

Draws a rectangle on the scene with the given width, height, color and thickness. Figure A.1 (2).

Properties: `Width` • `Height` • `TopLeft`, the 2D value of the top left corner of the rectangle (same value as `Position`) • `Color` • `Thickness` • `Fill`, determines if the rectangle must drawn filled.

Events: `SelectionEvent`.

A.1.3 Ellipse

Draws an ellipse on the scene with the given two radii. Figure A.1 (3).

Properties: `Center`, the 2D coordinates of the center of the ellipse (same value as `Position`) • `Radius1` • `Radius2` • `Color` • `Thickness` • `Fill`.

Events: `SelectionEvent`

A.1.4 Arrow

Draws an arrow with the given head and tail positions. The arrow has a text rendered at its tail. Figure A.1 (4).

Properties: `Head`, the 2D coordinate of the arrow head (same value as `Position`) • `Tail`, the 2D coordinates of the arrow tail • `TipLength`, the size of the arrow's tip • `Color` • `Thickness` • `Text`, the text that is rendered at the tail of the arrow • `TextFont`, the font for the rendered arrow text • `TextFontScale`, the size of the rendered text for arrow • `TextThickness`.

Events: `SelectionEvent`

A.1.5 Curve

Draws a curve as a list of 2D points on the scene. The application author defines the curve using the mouse by drawing it on the SceneViewer in the ISAR's authoring environment, or by using the tracked selection stick. A curve has a start and an end point, and its number of points can be defined. In the execution mode, when the application user moves the selection stick of the curve points `CurvePointHit` events are triggered. Figure A.1 (5).

Properties: `Start`, the 2D coordinate of the start point of the curve (same value as the `Position`) • `End`, the 2D coordinates of the end point of the curve • `NumOfPoints`, the number of points that are evenly distributed between the start and the end pints of the curve, along the path the application author drew • `PointColor`, the color to render the curve points • `PointHitColor`, curve points change their color to this value when they are hit, e.g. the selection stick passes over them • `PointRadius` each point of the curve is rendered as a circle with the give radius in pixels • `StartEndPointColor`, the color value to distinguish start and end point of the curve • `StartEndPointSize`, the radius value to distinguish start and end point of the curve.

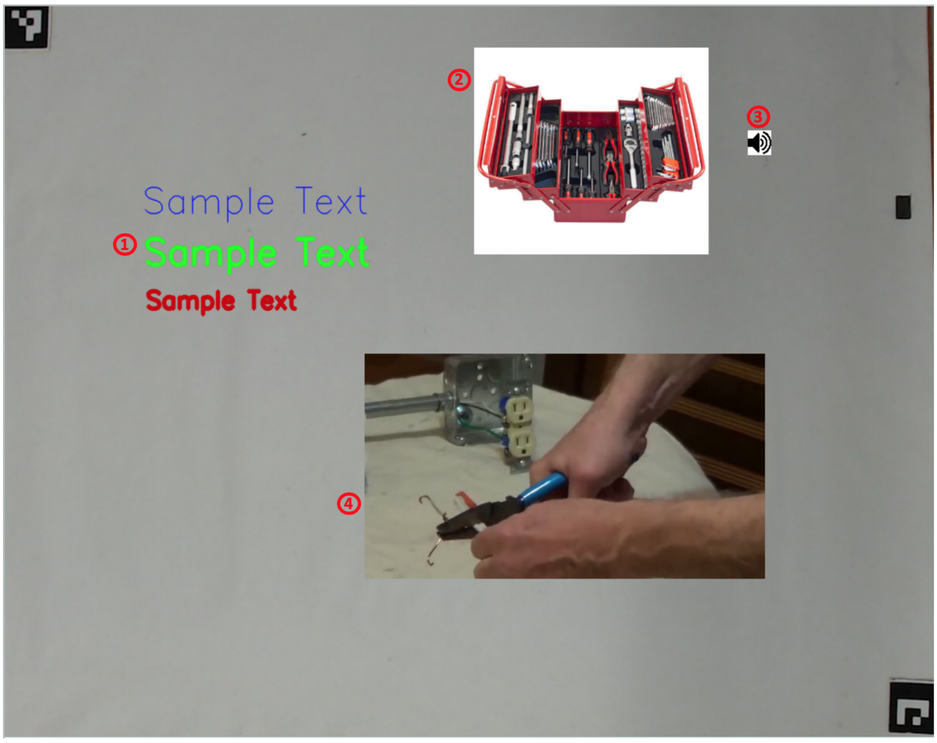


Fig. A.2 Multimedia annotations.

Events: `CurvePointHit`, fired when selection stick is over a curve point `CurveStartPointHit`, fired when selection stick is over the start point of the curve `CurveEndPointHit`, fired when the selection stick is over the end point of the curve.

A.2 Multimedia Annotations

A.2.1 Text

Renders a multi-line text with the give font and color. If the text length exceeds the `MaxLineWidth`, the text is wrapped to a new line. Figure A.2 (1).

Properties: `Position`, the bottom left corner of a rectangle fitting the text • `Text`, the text to render • `Color` • `Font` • `FontScale`, the font size • `Thickness` • `MaxLineWidth`, maximum line width to determine the position of line breaks.

A.2.2 Image

Renders an image on the scene with the give height and width. The image file must be in the project folder. Figure A.2 (2).

Properties: `Filename`, the name of the image file in the project folder • `Width` • `Height` • `KeepAspectRatio`, if true, then changing the height or width changes the other property to keep the image aspect ratio.

Events: `SelectionEvent`

A.2.3 Audio

Draws and icon for an audio file on the scene. The audio file must be in the project folder. If the icon is selected the audio is played back. The icon might be hidden. In that case the playback of the audio must be triggered using `StartAudio` action. Figure A.2 (3).

Properties: `Filename`, the name of the audio file in the project folder • `IconSize`, the size to render the audio icon • `Color`, the color to render the audio icon • `Text`, a text to render under the audio icon • `TextFont` • `TextFontScale` • `TextThickness` • `LoopPlayback`, if true, the playback of the audio is repeated until stopped.

Events: `SelectionEvent` • `AudioStarted`, is triggered as soon as audio playback is started • `AudioStopped`, is triggered as soon as audio playback is stopped.

Actions: `StartAudio`, starts the playback of the audio file • `StopAudio` stops the playback of the audio file.

A.2.4 Video

Renders a video file frame-by-frame on the scene with the given height and width. The video file must be in the project folder. If the video file contains audio, the audio is played back. In execution mode, if the user selects a video annotation with the selection stick, it start playback. If it is playing, the selection event pauses or un-pauses the playback. Figure A.2 (4).

Properties: `Filename`, the name of the video file in the project folder • `Width` • `Height` • `KeepAspectRatio`, if true, then changing the height or width change the other property to keep the video aspect ratio • `LoopPlayback`, if true, the video is repeated until

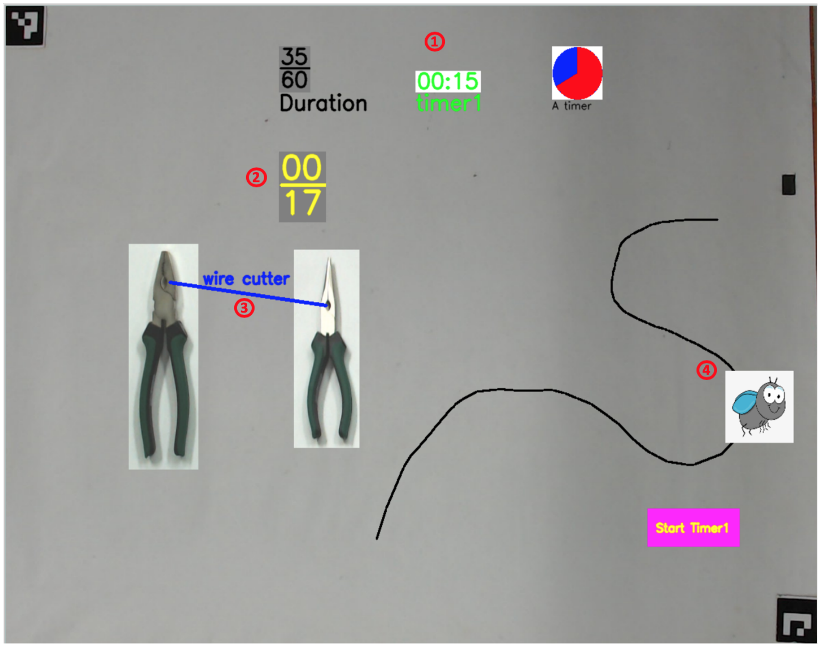


Fig. A.3 Dynamic annotations.

stopped • `Text`, a text to render under the video frame • `TextFont` • `TextFontScale` • `TextThickness`.

Events: `SelectionEvent` • `VideoStarted`, is triggered as soon as video playback is started • `VideoFinished`, is triggered as soon as video playback is finished.

Actions: `StartVideo`, starts the playback of the video file • `PauseVideo`, pauses the playback of the video file • `UnpauseVideo`, continues the playback of a paused video • `StopVideo` stops the playback of the video file.

A.3 Dynamic Annotations

A.3.1 Timer

Renders a timer on the scene. The timer has duration in and counts down till zero. The tick interval in seconds can be determined. Timer annotations be rendered differently: as chart that shows the elapsed and remaining time as a pie chart, as a normal countdown timer with minutes and seconds, and as a fraction. The timer has different timeout targets, upon which it triggers timeout events. Figure A.3 (1).

Properties: `Duration` the target value for the duration of the timer • `TickInterval`, the tick interval in seconds to decrement one from the duration • `Timeout1`, target value for the timeout one to trigger a timeout event • `Timeout2`, similar to timeout1 property • `Timeout3` similar to timeout1 property • `Text`, a text to be rendered under the timer • `TextFont`, the font for rendering timer value and text • `TextFontScale` • `TextThickness`.

Events: `TimerFinished`, triggered when timer has reached the target duration value • `Timeout1Reached`, triggered when the timer has reached timeout1 target value • `Timeout2Reached`, same as timeout1 reached event • `Timeout3Reached`, same as timeout1 reached event • `TimerTick`, triggered at each timer tick.

Actions: `StartTimer`, starts the timer • `StopTimer`, stops the timer • `ResetTimer`, resets the timer.

A.3.2 Counter

Renders a counter on the scene. The counter has a total number and can be incremented or decremented. The counter is rendered as a fraction showing the current value and the total number. The counter has also different target value, that when reached, it triggers respective events. Figure A.3 (2).

Properties: `Total Number`, the total target number of the counter • `Target1`, when target1 value is reached, the timer triggers a `Target1Reached` event • `Target2`, same as target1 property • `Target3`, same as target1 property • `Text`, a text to be rendered under the counter • `TextFont`, the font for rendering counter value and text • `TextFontScale` • `TextThickness`.

Events: `CounterIncremented`, triggered when the counter is incremented • `CounterDecremented`, triggered when the counter is decremented • `Target1Reached`, triggered when the counter has reached the value of target1 property • `Target2Reached`, similar to target1 reached event • `Target3Reached`, similar to target3 reached event •

Actions: `IncrementCounter`, increments the counter value • `DecrementCounter`, decrements the counter value • `ResetCounter`, sets the counter value to zero.

A.3.3 Relationship

A relationship annotation draws a labeled line between two physical objects. By default, the relationship annotation is only shown when both objects are on the tabletop. The application author can define if the relationship annotation is shown as soon as one of the objects is on the tabletop. Figure A.3 (3)

Properties: `Object1`, the first physical object for the relationship • `Position1`, the 2D position in the coordinate system of the first object • `Object2`, the other physical object of the relationship • `Position2`, the 2D position in the coordinate system of the second object • `LineThickness`, the thickness of the the connecting line • `LineColor`, the color of the connecting line • `Text`, the text of the label • `TextColor` • `TextFont` • `TextFontScale` • `TextThickness` • `TextPosition`, the distance from position1 along the line to place the text • `AlwaysShow`, if true the relationship line and label are shown as soon as one of the objects is placed on the table.

Events: `Object1Appeared`, triggered when the first object of the relationship is placed on the table • `Object1Removed`, triggered when the first object of the relationship is removed from the table • `Object2Appeared`, similar to object1 appeared event • `Object2Removed`, similar to object1 removed event.

A.3.4 Animation

An animation annotation renders an image moving along a given path with the given speed. The image file must be in the project folder. The path can be shown or hidden. The speed value determines how fast the position of the animation image is updated along the points of the path. The application author draws the path using mouse on the scene viewer in authoring environment. Figure A.3 (4).

Properties: `Image`, the name of the image filename in project folder • `ImageSize`, the scale factor to resize the image keeping the aspect ratio • `Speed`, the speed for moving the image along the animation path • `ShowPath`, if true the animation path is also rendered; otherwise the image moves along the invisible path • `PathColor`, the color of the path points if the path is shown • `PathThickness`, the thickness of the path if rendered • `Loop`, if true the image keeps moving between the start and and the end points until animation is stopped.

Events: `AnimationStarted`, triggered when the animation start • `AnimationFinished`, triggered when the animation stops.

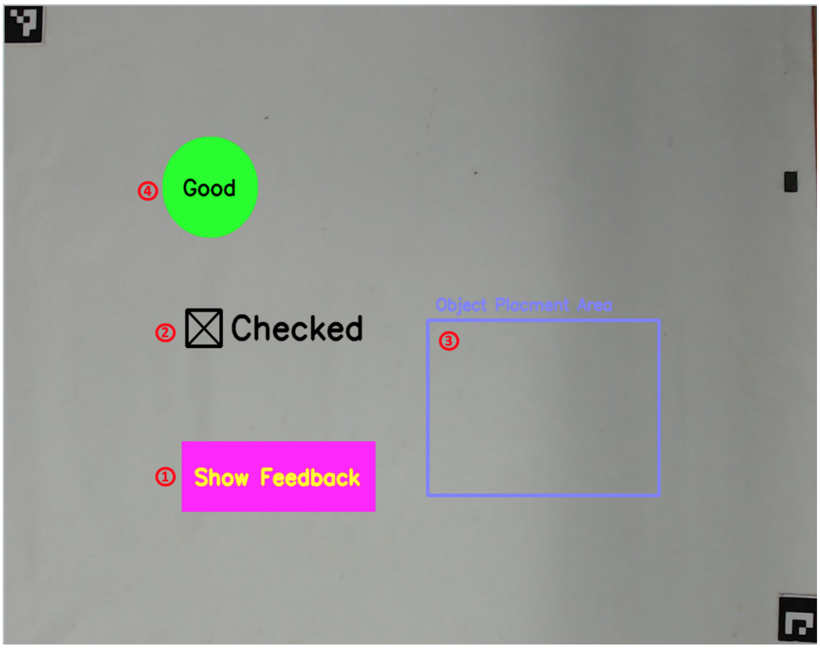


Fig. A.4 Interaction annotations.

Actions: `StartAnimation`, starts the animation • `StopAnimation`, stops the animation, setting the image position to the first point of the path • `PauseAnimation`, the animation is paused and the image stops moving • `ContinueAnimation`, continues a paused animation.

A.4 Interaction Annotations

A.4.1 ActionButton

Draws a rectangle button with text on the scene. An action can directly be defined using `OnSelect` property to be triggered as soon as the button is selected using selection stick or hand. Otherwise the `SelectionEvent` of the button can also be used to define actions to be triggered by the button as interaction rules. Figure A.4 (1).

Properties: `Width`, the width of the button's rectangle • `Height`, the height of the rectangle • `Color`, the color of the rectangle • `Thickness`, border thickness for the rectangle • `Fill`, if true the rectangle is rendered filled • `Text`, the text of the button • `TextColor` • `TextFont` • `TextFontScale` • `TextThickness` • `OnSelect`, the action to be performed when button is selected.

Events: `SelectionEvent`

A.4.2 Checkbox

Draws a checkbox, rendered as a checkable square with the given size, on the scene. A checkbox can also have a text that is rendered to its right side. The application user can toggle the check state by selecting the checkbox with the selection stick. The application author can also define checkbox groups by assigning each checkbox to a named group. Figure A.4 (2).

Properties: `Size`, the size of the checkbox square • `Color`, the color of the checkbox square and text • `Thickness` • `Text` • `TextFont` • `TextFontScale` • `TextThickness` • `Checked`, the initial check state of the checkbox • `Group`, the name of the checkbox group to which this checkbox belongs, or none.

Events: `CheckboxChecked`, triggered when the checkbox is checked • `CheckboxUnchecked`, triggered when the checkbox is unchecked • `CheckboxGroupChecked`, triggered when all the checkboxs of a checkbox group are selected • `CheckboxGroupUnchecked`, triggered when any of the checkboxs of a checkbox group is unchecked.

Actions: `Check`, sets the check stat of a checkbox to true • `Uncheck`, sets the check stat of a checkbox to false • `ToggleCheckbox`, toggles the check state of the checkbox.

A.4.3 ObjectPlacementArea

Using object placement area annotation the application author can define an area on the tabletop that triggers events when designated objects are placed inside it. An `ObjectPlacementAreaAnnotation` is simply rendered as rectangle. Figure A.4 (3).

Properties: `Width` • `Height` • `Color` • `Thickness` • `Fill`.

Events: `ObjectPlacedInArea`, triggered when the defined object is placed inside the object placement area • `ObjectRemovedFromArea`, triggered when the object is removed from the area.

A.4.4 Feedback

Feedback annotation shows a defined image and text and plays a defined audio file for three different feedback levels: good, bad, and average. For each level the application author can define the image, the text, and the audio file. The text can also be shown on a circle background. If image and audio is defined, the image and audio files must be inside the project folder. Figure A.4 (4).

Properties: `GoodImage`, the image for good feedback • `AverageImage`, the image for average feedback • `BadImage`, the image for the bad feedback • `ImageWidth` • `ImageHeight` • `GoodAudio`, the name of the audio file for good feedback • `AverageAudio`, the audio for average feedback • `BadAudio`, the audio file for bad feedback • `GoodText`, the text message for good feedback • `AverageText`, the text message for average feedback • `BadText`, the text message for bad feedback • `TextFont` • `TextFontScale` • `TextThickness` • `TextColor` • `ShowBackground`, if true the text is shown on a round background • `BackgroundColor`, the color for the background.

Actions: `ShowGoodFeedback`, shows the text and defined image for the good feedback and plays the audio if defined • `ShowAverageFeedback` • `ShowBadFeedback` •

Appendix B

Object Detection and Tracking

ISAR relies on object detection and feature matching for natural feature tracking. The tracking is model-based. Associated with each physical object is a *template image* that acts as the model for tracking. When the user (application author) adds a physical object to the scene, this template image is shown in scene viewer. For annotations that are attached to a physical object, the application author pre-registers them according to this template model. The attached annotations are rendered with reference to the coordinate system of the physical object, which corresponds to the (0, 0) in template image. For markerless tracking, the objects are first detected using a trained YOLOv3 model, that returns the bounding box of the objects in the camera image. The object bounding boxes are cropped from the camera image and the 2D pose of the objects is determined in correspondence to the template image using feature matching. For marker-based tracking object detection and tracking is based on ArUco [RRMSMC18] markers.

Markerless or marker-based object detection and tracking plugins are copied into the plugins folder of ISAR. Each object detection and tracking plugin must have the following structure:

- Plugin descriptor and interface module: Each object detection and tracking plugin must include an `objectdetector.py` module. The object detection and tracking service of ISAR looks for this file in the plugin folder and loads it. This file has the standard interface for object detection and tracking. It consists of the `get_predictions()` method that receives a camera frame and returns the result of object detection and tracking for it (see below), the name and description of the plugin, the `get_physical_objects()` method that returns a list of all physical objects that can be detected and tracked using this plugin, and the `terminate()` method that signals the plugin to terminate.

- A `physical_objects.json` file that contains a list of physical objects that this object detection and tracking plugin can detect and track. This file contains the name and the path to the template image for each of the physical objects.
- A folder containing the template images. One for each physical object.
- Any other files needed for object detection and tracking, for example the CNN model in case of YOLO object detection or the marker associations in case of marker-based object recognition and tracking.

ISAR object detection and tracking service does not depend on how an object detection and tracking plugin tracks objects. The plugin only provides a standard interface, the `get_predictions()` method, that receives an `ObjectDetectionRequest` instance. The `ObjectDetectionRequest` instance contains the camera frame, for which the object detection and tracking must be performed. It also contains a list of physical objects available in the scene. To achieve a better performance, specially in case of markerless tracking, the pose is only estimated for the objects that are available in the scene and not every object that is in the camera image. The `get_predictions()` method return a list of `ObjectDetectionPrediction` instances. An `ObjectDetectionPrediction` instance contains the `lablel` of the detected object, the detection `confidence`, the `bounding_box`, the `cropped_image` of the detected object (the camera image cropped at the bounding box of the detected object), and the `pose` which is the 3x2 2D affine transformation matrix.

The object detection model returns the bounding box of the physical objects in the camera image. For each object the camera image is cropped at this bounding box to get the current image of the physical object as it is on the tabletop. We call this image target image. The tracking algorithm calculates the affine 2D transformation that aligns the template image of the physical object to this cropped target image. For each camera frame, the object detection and tracking algorithm works as follows:

1. Get the bounding box of physical objects in the camera image from YOLO model. For each physical object crop the camera image at the bounding box to get the current target image of the physical object.
2. Extract key points for template image and target image using AKAZE [AS11].
3. Find the matching key points using Hamming distance.
4. Estimate an initial affine 2D transformation from key points using RANSAC [FB81] algorithm. Then calculate a refined transformation using only the inliers with LMedS [MKRL86] algorithm.

5. Calculate the final refined pose, using the estimated transformation from previous step, with the ECC algorithm [EP08].

In order to make the pose estimation results more robust and stable, the pose estimation algorithm maintains and updates a dictionary that contains the best estimated pose for each physical object in the scene. Every time the algorithm is performed, the result of estimated pose is compared to the best estimated transformation so far. For the comparison, the template image is warped using both transformation matrices and the respective error in relation to the target image is calculated. The errors is calculated using the root mean squared error of the difference of between warped image and target image. If the new transformation is better than the existing best transformation, it is returned and also replaces the best transformation. Otherwise the best transformation is returned.

Appendix C

Extending ISAR Framework

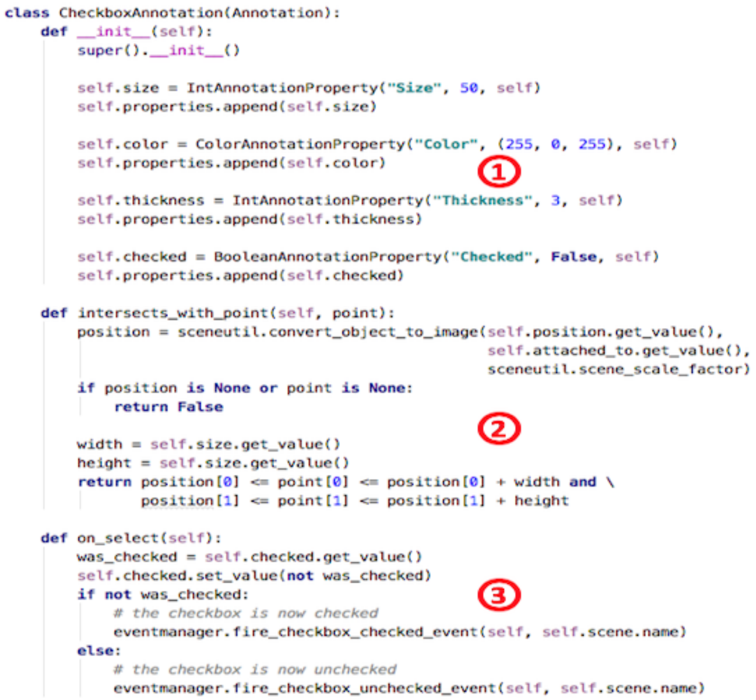
ISAR framework and authoring environment can be extended to meet new requirements. A framework extender can added new functionality to ISAR in three ways: adding new annotation types, adding new action types, and adding new event types.

C.1 Adding New Annotation Types

All annotations in ISAR are subclass of the abstract base class `Annotation`. The base class Annotation provides basic properties and functionalities common to all annotations. This includes properties such as name, id, and position and functionalities such as attaching to physical objects or deleting the annotations from the scene.

Each new annotation type must also define its corresponding annotations tool class that is responsible for rendering the annotation. All annotation tool classes are subclasses of the abstract base class `AnnotationTool`. An annotation tool has drawing mode that is active when the user is defining the annotation. In this mode the annotation tool responses to mouse events (mouse pressed, mouse moved, and mouse released) to preview the annotation and added to the scene. When the drawing mode is inactive, the annotation tool renders the annotation according to its defined properties.

In order to add a new annotation type to ISAR the framework extender must do the following steps. We illustrate the steps on the example of the `CheckboxAnnotation`:



```python
class CheckboxAnnotation(Annotation):
    def __init__(self):
        super().__init__()

        self.size = IntAnnotationProperty("Size", 50, self)
        self.properties.append(self.size)

        self.color = ColorAnnotationProperty("Color", (255, 0, 255), self)
        self.properties.append(self.color)

        self.thickness = IntAnnotationProperty("Thickness", 3, self)
        self.properties.append(self.thickness)

        self.checked = BooleanAnnotationProperty("Checked", False, self)
        self.properties.append(self.checked)

    def intersects_with_point(self, point):
        position = sceneutil.convert_object_to_image(self.position.get_value(),
                                                     self.attached_to.get_value(),
                                                     sceneutil.scene_scale_factor)
        if position is None or point is None:
            return False

        width = self.size.get_value()
        height = self.size.get_value()
        return position[0] <= point[0] <= position[0] + width and \
               position[1] <= point[1] <= position[1] + height

    def on_select(self):
        was_checked = self.checked.get_value()
        self.checked.set_value(not was_checked)
        if not was_checked:
            # the checkbox is now checked
            eventmanager.fire_checkbox_checked_event(self, self.scene.name)
        else:
            # the checkbox is now unchecked
            eventmanager.fire_checkbox_unchecked_event(self, self.scene.name)
```

Fig. C.1 Example of how to define the class for a new annotation type.

Adding the annotation class:

The framework extender must add a class for the new annotation that inherits from the Annotation base class. He should also define any additional properties for the annotation using the different annotation property classes. In the case of CheckboxAnnotation these are the Size, an int value for defining the size of the checkbox rectangle in pixels, the Color for the border color of the checkbox rectangle, the Thickness an int value for the border thickness of the checkbox rectangle, and the Checked, a boolean property for the initial check state of the checkbox (figure C.1 [1]).

The framework extender should also implement any necessary basic functionality of the annotation by overriding the methods of the Annotation base class. For example in the case of CheckboxAnnotation, the intersects_with_point() method should be implemented, that checks if a given point is withing the boundaries of the checkbox rectangle (figure C.1 [2]). This method is used for example by selection service in order to detect selection events on the annotation.

The framework extender should also implement necessary functionality to fire any events related to the annotation. In the case of CheckboxAnnotation these are checkbox checked and checkbox unchecked events (figure C.1 [3]).

Adding annotation tool class:

For each new annotation type a corresponding annotation tool class must also be added that inherits for the abstract base class `AnnotationTool`. The annotation tool class is responsible for rendering the annotation instance, both as it is being drawn and after it is finished and added to the scene. In the latter case, the scene renderer delegate the drawing of all annotations in the scene to the corresponding annotation tool class. In the former case the annotation tool class receives mouse events from scene viewer in authoring UI.

The annotation instance acts as the single source of truth. During the drawing, the mouse interactions change the properties of a temporarily created annotation instance. At mouse release event, the validity of the annotation is checked and it is added to the scene. The annotation tool uses the annotation instance to render it as both as being drawn as it is finished and added the scene.

For each annotation instance, the scene renderer delegates its drawing to the corresponding annotation tool class. So, the annotation class must be related to the annotation tool class. This is done by adding an entry for the `annotation_tools` dictionary that maps the name of the annotation class to the corresponding instance of annotation tool.

Adding annotation tool to the user interface of authoring environment:

Finally, a button must be added for the new annotation type to the annotation tools toolbar in the graphical user interfaces of the authoring environment. The framework extender must connect that button to the corresponding annotation tool class by adding it to the `annotation_tool_btns` dictionary. This dictionary maps the button name in the user interface to the annotation tool class, so that whenever the button is clicked, the active annotation tool is set to the corresponding annotation tool.

C.2 Adding New Event Types

All event classes are subclasses of the abstract base class `Event`, that defines common properties and functionalities of all events, such as name, scene, and event sources. Each event type has an array of possible source types, for example `SelectionEvent` has physical objects and annotations as possible sources. All subclasses of a source type can also be selected as source type for the event. The `Event` superclass also define the `equals()` method that checks if two event instances are equal. This is needed for firing rules, because the RuleSevice fires a rule, whenever its corresponding event has occurred.

In order to add a new event type the framework extender must do the following:

1. Add a class for the event type that inherits for the abstract base class Event, and define all the necessary properties for the event type. For example, the SelectionEvent has a property for trigger_interval, that defines the delay before the section event is fired.

2. Define the array of source types for the event. All subclasses of a source type can also be selected as source type for the event. For example SelectionEvent has physical object and annotation as its source type. Accordingly, all physical objects and annotation instances on the scene can be selected as its possible sources.

3. If the event type has any properties, implement the corresponding user interface code in the abstract method update_event_properties_frame() that allows setting those properties. For example, the PhysicalObjectPlacedInAreaEvent has an instance of a ObjectPlacementAreaAnnotation as its source and the physical object as its property. It provides the corresponding UI for selecting the physical object in the event definition dialog.

4. Add the event class to the list of events, so that the user can select it in the event definition dialog.

C.3 Adding New Action Types

All action classes are subclasses of the abstract base class Action. The base class Action defines basic properties and functionality of all action types, such as name and scene, and checking the validity of action targets. Each action type has an array of possible target types.

In order to add a new action type the framework extender must do the following:

1. Add a class for the action type that inherits for the abstract base class Action.

2. Define the array of target types for the action. For each target type, all its subclasses can also be used as target types of the action instance. For example the ToggleAnnotationVisibilityAction has target type Annotation, which makes all annotation instances possible targets of this action type.

3. Define if the action is bound to a scene or global. Global actions, such as `ShowSceneAction` do not need a scene. All other actions belong to a specific scene and act only on the targets that exist on that scene.
4. If the action has any properties, implement the corresponding user interface code in the abstract method `update_action_properties_frame()` that allows setting those properties. For example, the `HighlightPhysicalObjectAction` has an instance of a physical object as its target and the color of highlighting as a property. It provides the corresponding UI for setting the color property in the action definition dialog.
5. Implement the abstract `run()` method. The run method is called whenever the action must be run, for example when a rule is fired that has this action instance. The `run()` method implements how the action affects its targets.
6. Add the action class to the list of actions, so that the user can select it in the action definition dialog.

Appendix D

User Study

Dear Participant,

thank you for taking part in this study. The study is structured in four parts:

1. I first ask you some questions about your experience in software development
2. I then do a short demonstration of ISAR and its features by creating a simple application
3. You then have 1-hour time to create your own ISAR application for vocabulary learning.
4. I finally ask you some questions about usability of ISAR and its effectiveness.

Part 1: Development Experience

1.1. What is your age?

under 20 20-30 30-40 40-50 over 50

1.2. How technically affine are you?

- I am not technically affine. [low]
- I can connect and setup my home printer and home cinema system. [moderate]
- I am technically affine. I like testing the functions of new technical systems. When I have a new technical system in front of me, I try it out intensively. [high]
- I predominantly deal with technical systems as part of my job. [very high]

1.3. What is your experience with programming?

- I don't have any experience with programming. [no experience]
- I can write simple toy programs for fun or homework assignments. [beginner]
- I have under 2 years of programming experience as part of my job. I have programmed system with less than 5000 lines of code. [advanced]
- I have more than 2 years programming experience as part of my job. I have programmed large systems with more that 5000 lines of code [expert]

1.4. How familiar are you with augmented reality (AR)?

- I don't know what is augmented reality. I have never used an augmented reality system. [not familiar]
- I have tested or used AR systems, but I don't know how they are created. [only used]
- I have created my own AR systems using frameworks like ARKit, ARToolkit, Vuforia, ... [beginner developer]
- I have developed my own AR systems from scratch. [expert]

1.5. What are the main components of an AR system? What are the main challenges in developing an AR system?

Part 2: ISAR Demonstration [~ 15 Minutes]

I now demonstrate different features of ISAR that you need (or can use) in order to create your vocabulary learning application. You see a demonstration of the following features of ISAR:

- Creating a project
- Creating scenes
- Adding annotations to the scene
- Configuring annotation properties
- Adding physical objects to the scene
- Defining scene navigation
- Defining events, actions, and interaction rules
- Selecting annotations and physical objects

Part 3: Authoring an ISAR application for vocabulary learning [at most 60 minutes]

Now you will use ISAR to develop an application for learning vocabulary related to tools from a normal household toolbox. You are free to design your ISAR application the way you like. But your application must have at least the following features:

- At least two scenes with scene navigation
- Each scene must contain one or more physical objects and one or more annotations
- The scenes should help the application user learn the vocabulary related to the tools, for example name of the tools, related verbs, name of the related objects, etc.
- Each scene must have defined interaction rule, for example “Selecting a tool plays back the audio annotation for its name”

The object recognition and tracking package is provided to you, so you don’t have to create your own object recognition and tracking package for this application. Sample media files (images, audio, video) related to each tool are also provided to you. You don’t have to use them though, and you can create your own media files or download them from internet.

Please think aloud and explain what you are doing as you create and test your application.

Part 4: Usability Questions

4.1. How long do you think it would take to implement the application you created from scratch?

4.2. Which features of ISAR do you find most useful regarding development of an interactive augmented reality application for tabletop?

4.3. What were the main limitations and difficulties you faced in creating your application?

4.4. Which application domains do you think would be the most suitable for using ISAR?

4.5. I think users without technical background can use ISAR to create their own interactive applications for the tabletop (after a short, e.g. one-hour, introduction session).

strongly agree   agree   neutral   disagree   strongly disagree

4.6. As an expert of other domains (e.g. a teacher) I would use ISAR to create my own interactive applications.

strongly agree   agree   neutral   disagree   strongly disagree

4.7. I would introduce and recommend ISAR to experts of other domains.

strongly agree   agree   neutral   disagree   strongly disagree

System Usability Scale Questionnaire:

1. **I think that I would like to use this system frequently**

strongly agree agree neutral disagree strongly disagree

2. **I found the system unnecessarily complex.**

strongly agree agree neutral disagree strongly disagree

3. **I thought the system was easy to use.**

strongly agree agree neutral disagree strongly disagree

4. **I think that I would need the support of a technical person to be able to use this system.**

strongly agree agree neutral disagree strongly disagree

5. **I found the various functions in this system were well integrated.**

strongly agree agree neutral disagree strongly disagree

6. **I thought there was too much inconsistency in this system.**

strongly agree agree neutral disagree strongly disagree

7. **I would imagine that most people would learn to use this system very quickly.**

strongly agree agree neutral disagree strongly disagree

8. **I found the system very cumbersome to use.**

strongly agree agree neutral disagree strongly disagree

9. **I felt very confident using the system.**

strongly agree agree neutral disagree strongly disagree

10. **I needed to learn a lot of things before I could get going with this system.**

strongly agree agree neutral disagree strongly disagree

References

[AAG$^+$09] Michelle Annett, Fraser Anderson, Darrell Goertzen, Jonathan Halton, Quentin Ranson, Walter F Bischof, and Pierre Boulanger. Using a multi-touch tabletop for upper extremity motor rehabilitation. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, pages 261–264. ACM, 2009.

[Ack01] Edith Ackermann. Piaget's constructivism, papert's constructionism: What's the difference. *Future of learning group publication*, 5(3):438, 2001.

[ADG04] Daniel F Abawi, Ralf Dörner, and Paul Grimm. A component-based authoring environment for creating multimedia-rich mixed reality. In *Proceedings of the Seventh Eurographics conference on Multimedia*, pages 31–40, 2004.

[Ain99] Shaaron Ainsworth. The functions of multiple representations. *Computers & education*, 33(2-3):131–152, 1999.

[AIRH12] Hussain Al-Issa, Holger Regenbrecht, and Leigh Hale. Augmented reality applications in rehabilitation to improve physical outcomes. *Physical Therapy Reviews*, 17(1):16–28, 2012.

[All16] Thomas Allweyer. *BPMN 2.0: introduction to the standard for business process modeling*. BoD–Books on Demand, 2016.

[ANRS$^+$13] Mirjam Augstein, Thomas Neumayr, Renate Ruckser-Scherb, Isabel Karlhuber, and Josef Altmann. The fun. tast. tisch. project: a novel approach to neuro-rehabilitation using an interactive multiuser multitouch tabletop. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*, pages 81–90. ACM, 2013.

[AS11] Pablo F Alcantarilla and T Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7):1281–1298, 2011.

[ATH$^+$18] Yuliana Apaza, Richard Tumailla, Wilder Hancco, Alfredo Paz-Valderrama, Carlo Corrales-Delgado, and Manuel Loaiza. Systematic mapping study on high-level content design frameworks for augmented reality. In *2018 20th Symposium on Virtual and Augmented Reality (SVR)*, pages 192–201. IEEE, 2018.

[Azu97] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.

[BBK$^+$01] Martin Bauer, Bernd Bruegge, Gudrun Klinker, Asa MacWilliams, Thomas Reicher, Stefan Riss, Christian Sandor, and Martin Wagner. Design of a component-based augmented reality framework. In *Proceedings IEEE and ACM International Symposium on Augmented Reality*, pages 45–54. IEEE, 2001.

[BCFP19] Barbara Rita Barricelli, Fabio Cassano, Daniela Fogli, and Antonio Piccinno. End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137, 2019.

[BCL$^+$15] Mark Billinghurst, Adrian Clark, Gun Lee, et al. A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272, 2015.

[BD03] Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering: Using UML, Patterns and Java, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[BDOB13] Engin Bumbacher, Amit Deutsch, Nancy Otero, and Paulo Blikstein. Beattable: a tangible approach to rhythms and ratios. In *Proceedings of the 12th International Conference on Interaction Design and Children*, pages 589–592, 2013.

[BF02] Oliver Bimber and Bemd Frohlich. Occlusion shadows: Using projected light to generate realistic occlusion effects for view-dependent optical see-through displays. In *Proceedings. International Symposium on Mixed and Augmented Reality*, pages 186–319. IEEE, 2002.

[BFSR16] Sebastian Büttner, Markus Funk, Oliver Sand, and Carsten Röcker. Using head-mounted displays and in-situ projection for assistive systems: A comparison. In *Proceedings of the 9th ACM international conference on pervasive technologies related to assistive environments*, pages 1–8, 2016.

[BGW$^+$02] Oliver Bimber, Stephen M Gatesy, Lawrence M Witmer, Ramesh Raskar, and L Miguel Encarnação. Merging fossil specimens with computer-generated information. *Computer*, (9):25–30, 2002.

[BK00] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb's journal of software tools*, 3, 2000.

[BKBN12] Tobias Blum, Valerie Kleeberger, Christoph Bichlmeier, and Nassir Navab. mirracle: An augmented reality magic mirror system for anatomy education. In *2012 IEEE Virtual Reality Workshops (VRW)*, pages 115–116. IEEE, 2012.

[BKM08] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BKM09] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[BMBC19] Clara Bonillo, Javier Marco, Sandra Baldassarri, and Eva Cerezo. Kitvision toolkit: supporting the creation of cognitive activities for tangible tabletop devices. *Universal Access in the Information Society*, pages 1–29, 2019.

[BMF$^+$17] Sebastian Büttner, Henrik Mucha, Markus Funk, Thomas Kosch, Mario Aehnelt, Sebastian Robert, and Carsten Röcker. The design space of augmented and virtual reality applications for assistive environments in manufacturing: a visual approach. In *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*, pages 433–440, 2017.

[BR05] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality: merging real and virtual worlds*. AK Peters/CRC Press, 2005.

[BRF01] Deepak Bandyopadhyay, Ramesh Raskar, and Henry Fuchs. Dynamic shader lamps: Painting on movable objects. In *Proceedings IEEE and ACM International Symposium on Augmented Reality*, pages 207–216. IEEE, 2001.

[BW19] Bhaskar Bhattacharya and Eliot H Winer. Augmented reality via expert demonstration authoring (areda). *Computers in Industry*, 105:61–79, 2019.

[CDZOD15] Sébastien Cuendet, Jessica Dehler-Zufferey, Giulia Ortoleva, and Pierre Dillenbourg. An integrated way of using a tangible user interface in a classroom. *International Journal of Computer-Supported Collaborative Learning*, 10(2):183–208, 2015.

[Chi16] Pei-Yu Chi. *Designing Video-Based Interactive Instructions*. PhD thesis, UC Berkeley, 2016.

[CKL$^+$10] Jinhyuk Choi, Youngsun Kim, Myonghee Lee, Gerard J Kim, Yanghee Nam, and Yongmoo Kwon. k-mart: Authoring tool for mixed reality contents. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 219–220. IEEE, 2010.

[CLL$^+$13] Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Bjoern Hartmann. Democut: generating concise instructional videos for physical demonstrations. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 141–150, 2013.

[CMK88] John M Carroll, Robert L Mack, and Wendy A Kellogg. Interface metaphors and user interface design. In *Handbook of human-computer interaction*, pages 67–85. Elsevier, 1988.

[CRR08] Nadine Couture, Guillaume Rivière, and Patrick Reuter. Geotui: a tangible user interface for geoscience. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 89–96, 2008.

[DBLS18] Arindam Dey, Mark Billinghurst, Robert W Lindeman, and J Swan. A systematic review of 10 years of augmented reality usability studies: 2005 to 2014. *Frontiers in Robotics and AI*, 5:37, 2018.

[DD14] Matt Dunleavy and Chris Dede. Augmented reality teaching and learning. In *Handbook of research on educational communications and technology*, pages 735–745. Springer, 2014.

[DDLÓ$^+$10] Alan Dunne, Son Do-Lenh, Gearóid Ó'Laighin, Chia Shen, and Paolo Bonato. Upper extremity rehabilitation of children with cerebral palsy using accelerometer feedback on a multitouch display. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 1751–1754. IEEE, 2010.

[DE11] Pierre Dillenbourg and Michael Evans. Interactive tabletops in education. *International Journal of Computer-Supported Collaborative Learning*, 6(4):491–514, 2011.

[DGHP03] Ralf Dörner, Christian Geiger, Michael Haller, and Volker Paelke. Authoring mixed reality—a component and framework-based approach. In *Entertainment Computing*, pages 405–413. Springer, 2003.

[DLKD09] Son Do-Lenh, Frédéric Kaplan, and Pierre Dillenbourg. based concept map: The effects of tabletop on an expressive collaborative learning task. In *The 23rd BCS conference on Human Computer Interaction (HCI 2009)*, number CONF, pages 149–158. ACM, 2009.

[Do12] Lenh Hung Son Do. *Supporting reflection and classroom orchestration with tangible tabletops*. PhD thesis, EPFL, 2012.

[Dou04] Paul Dourish. *Where the action is: the foundations of embodied interaction*. MIT press, 2004.

[DRSG18] Clifford De Raffaele, Serengul Smith, and Orhan Gemikonakli. An active tangible user interface framework for teaching and learning artificial intelligence. In *23rd International Conference on Intelligent User Interfaces*, pages 535–546, 2018.

[EB09] Darren Edge and Alan F Blackwell. Peripheral tangible interaction by analytic design. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 69–76, 2009.

[Ech09] Florian Echtler. *Tangible information displays*. PhD thesis, Technische Universität München, 2009.

[Edw95] Laurie D Edwards. The design and analysis of a mathematical microworld. *Journal of Educational Computing Research*, 12(1):77–94, 1995.

[EP08] Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008.

[ER14] Michael A Evans and Jochen Rick. Supporting learning with interactive surfaces and spaces. In *Handbook of research on educational communications and technology*, pages 689–701. Springer, 2014.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FBÅM$^+$14] Åsa Fast-Berglund, Magnus Åkerman, Sandra Mattsson, Pierre EC Johansson, Anna Malm, and Anna Perenstål Brenden. Creating strategies for global assembly instructions–current state analysis. In *The sixth Swedish Production Symposium*, 2014.

[FBR97] Morten Fjeld, Martin Bichsel, and Matthias Rauterberg. Build-it: an intuitive design tool based on direct object manipulation. In *International Gesture Workshop*, pages 297–308. Springer, 1997.

[FIB95] George W Fitzmaurice, Hiroshi Ishii, and William AS Buxton. Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449, 1995.

[Fis04] Kenneth P Fishkin. A taxonomy for and analysis of tangible interfaces. *Personal and Ubiquitous computing*, 8(5):347–358, 2004.

[Fis13] Gerhard Fischer. End-user development: from creating technologies to transforming cultures. In *International Symposium on End User Development*, pages 217–222. Springer, 2013.

[FKS16] Markus Funk, Thomas Kosch, and Albrecht Schmidt. Interactive worker assistance: comparing the effects of in-situ projection, head-mounted displays, tablet, and paper instructions. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 934–939, 2016.

[FLM$^+$18] Markus Funk, Lars Lischke, Sven Mayer, Alireza Sahami Shirazi, and Albrecht Schmidt. Teach me how! interactive assembly instructions using demonstration and in-situ projection. In *Assistive Augmentation*, pages 49–73. Springer, 2018.

[FSM+15] Markus Funk, Alireza Sahami Shirazi, Sven Mayer, Lars Lischke, and Albrecht Schmidt. Pick from here! an interactive mobile cart using in-situ projection for order picking. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 601–609, 2015.

[Gam95] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[GB08] Saul Greenberg and Bill Buxton. Usability evaluation considered harmful (some of the time). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 111–120. ACM, 2008.

[GJMSMCMJ14] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[GLC15] Pavel Gurevich, Joel Lanir, and Benjamin Cohen. Design and implementation of teleadvisor: a projection-based augmented reality system for remote collaboration. *Computer Supported Cooperative Work (CSCW)*, 24(6):527–562, 2015.

[GLG11] Jens Grubert, Tobias Langlotz, and Raphaël Grasset. Augmented reality browser survey. *Tehnical report, Institute for Computer Graphics and Vision, Graz University of Technology, Austria*, 2011.

[GMOF13] Jesús Gimeno, Pedro Morillo, Juan Manuel Orduña, and Marcos Fernández. A new ar authoring tool using depth maps for industrial procedures. *Computers in Industry*, 64(9):1263–1271, 2013.

[GNTH14] Steffen Gauglitz, Benjamin Nuernberger, Matthew Turk, and Tobias Höllerer. In touch with the remote world: Remote collaboration with augmented reality drawings and virtual navigation. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pages 197–205. ACM, 2014.

[GSH+07] Audrey Girouard, Erin Treacy Solovey, Leanne M Hirshfield, Stacey Ecott, Orit Shaer, and Robert JK Jacob. Smart blocks: a tangible mathematical manipulative. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 183–186, 2007.

[GTOF12] Jesús Gimeno, PM Tena, Juan M Orduna, and Marcos Fernández. An advanced authoring tool for augmented reality applications in industry. *Actas de las XXIII Jornadas de Paralelismo (JP 2012). Elche: Servicio de Publicaciones de la Universidad Miguel Hernández*, 2012.

[HA12] John Hardy and Jason Alexander. Toolkit support for interactive projected displays. In *Proceedings of the 11th international conference on mobile and ubiquitous multimedia*, pages 1–10, 2012.

[Har14] John Hardy. *Toolkit support for interactive projected displays*. PhD thesis, Lancaster University, 2014.

[Hei91] George Hein. Constructivist learning theory. *Institute for Inquiry. Available at:/http://www. exploratorium. edu/ifi/resources/constructivistlearning. htmlS*, 1991.

[HKD$^+$13] Hossein Mousavi Hondori, Maryam Khademi, Lucy Dodakian, Steven C Cramer, and Cristina Videira Lopes. A spatial augmented reality rehab system for post-stroke hand rehabilitation. In *MMVR*, volume 184, pages 279–285, 2013.

[HR02] Matthias Haringer and Holger T Regenbrecht. A pragmatic approach to augmented reality authoring. In *Proceedings. International Symposium on Mixed and Augmented Reality*, pages 237–245. IEEE, 2002.

[HRL99] Lars Erik Holmquist, Johan Redström, and Peter Ljungstrand. Token-based access to digital information. In *International Symposium on Handheld and Ubiquitous Computing*, pages 234–245. Springer, 1999.

[HSCJ09] Michael S Horn, Erin Treacy Solovey, R Jordan Crouser, and Robert JK Jacob. Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 975–984, 2009.

[HSGB06] Alastair Hampshire, Hartmut Seichter, Raphaël Grasset, and Mark Billinghurst. Augmented reality authoring: generic context from programmer to designer. In *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*, pages 409–412, 2006.

[HSZ05] Michael Haller, Erwin Stauder, and Juergen Zauner. Amire-es: Authoring mixed reality once, run it anywhere. In *Proceedings of the 11th International Conference on Human-Computer Interaction (HCII)*, volume 2005, 2005.

[HWL$^+$10] Taejin Ha, Woontack Woo, Youngho Lee, Junhun Lee, Jeha Ryu, Hankyun Choi, and Kwanheng Lee. Artalet: tangible user interface based immersive augmented reality authoring tool for digilog book. In *2010 International Symposium on Ubiquitous Virtual Reality*, pages 40–43. IEEE, 2010.

[Ish08] Hiroshi Ishii. The tangible user interface and its evolution. *Communications of the ACM*, 51(6):32–36, 2008.

[IU97] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 234–241, 1997.

[JGAK07] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reactable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 139–146, 2007.

[JGH$^+$08] Robert JK Jacob, Audrey Girouard, Leanne M Hirshfield, Michael S Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: a framework for post-wimp interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 201–210, 2008.

[JHJL01] Wendy Ju, Rebecca Hurwitz, Tilke Judd, and Bonny Lee. Counteractive: an interactive cookbook for the kitchen counter. In *CHI'01 extended abstracts on Human factors in computing systems*, pages 269–270, 2001.

[JIPP02] Robert JK Jacob, Hiroshi Ishii, Gian Pangaro, and James Patten. A tangible interface for organizing information using a grid. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 339–346, 2002.

[Kal09] Martin Kaltenbrunner. reactivision and tuio: a tangible tabletop toolkit. In *Proceedings of the ACM international Conference on interactive Tabletops and Surfaces*, pages 9–16, 2009.

[KB99] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94. IEEE, 1999.

[KB07] Martin Kaltenbrunner and Ross Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 69–74, 2007.

[KBP$^+$00] Hirokazu Kato, Mark Billinghurst, Ivan Poupyrev, Kenji Imamoto, and Keihachiro Tachibana. Virtual object manipulation on a table-top ar environment. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 111–119. Ieee, 2000.

[KF10] Andreas Kunz and Morten Fjeld. From table–system to tabletop: Integrating technology into interactive surfaces. In *Tabletops-Horizontal Interactive Displays*, pages 51–69. Springer, 2010.

[KFA$^+$14] Oliver Korn, Markus Funk, Stephan Abele, Thomas Hörz, and Albrecht Schmidt. Context-aware assistive systems at the workplace: analyzing the effects of projection and gamification. In *Proceedings of the 7th international conference on pervasive technologies related to assistive environments*, pages 1–8, 2014.

[KHT06] Scott R Klemmer, Björn Hartmann, and Leila Takayama. How bodies matter: five themes for interaction design. In *Proceedings of the 6th conference on Designing Interactive systems*, pages 140–149, 2006.

[KIR95] D KIRSH. The intelligent use of space. *Artificial Intelligence*, 73:31–68, 1995.

[KLLL04] Scott R Klemmer, Jack Li, James Lin, and James A Landay. Papier-mache: toolkit support for tangible input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 399–406, 2004.

[KM08] Mi Jeong Kim and Mary Lou Maher. The impact of tangible user interfaces on designers' spatial cognition. *Human–Computer Interaction*, 23(2):101–137, 2008.

[KNG16] Sunwook Kim, Maury A Nussbaum, and Joseph L Gabbard. Augmented reality "smart glasses" in the workplace: industry perspectives and challenges for worker safety and health. *IIE transactions on occupational ergonomics and human factors*, 4(4):253–258, 2016.

[KP$^+$88] Glenn E Krasner, Stephen T Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[KS03] Toshikazu Karitsuka and Kosuke Sato. A wearable mixed reality with an on-board projector. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 321–322. IEEE, 2003.

[KSF06] David Kirk and Danae Stanton Fraser. Comparing remote gesture technologies for supporting collaborative physical tasks. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1191–1200. ACM, 2006.

[KWCS05] Christian Knopfle, Jens Weidenhausen, Laurent Chauvigne, and Ingo Stock. Template based authoring for ar based service scenarios. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pages 237–240. IEEE, 2005.

[LBH$^+$10] Johannes Luderschmidt, Immanuel Bauer, Nadia Haubner, Simon Lehmann, Ralf Dörner, and Ulrich Schwanecke. Tuio as3: A multi-touch and tangible user interface rapid prototyping toolkit for tabletop interaction. In *Self Integrating Systems for Better Living Environments: First Workshop, Sensyble*, pages 21–28, 2010.

[LH12] Jérémy Laviole and Martin Hachet. Papart: interactive 3d graphics and multi-touch augmented paper for artistic creation. In *2012 IEEE symposium on 3D user interfaces (3DUI)*, pages 3–6. IEEE, 2012.

[LHLD13] Johannes Luderschmidt, Nadia Haubner, Simon Lehmann, and Ralf Dörner. Emil: a rapid prototyping authoring environment for the design of interactive surface applications. In *International Conference on Human-Computer Interaction*, pages 381–390. Springer, 2013.

[LHV+18] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. Evaluation strategies for hci toolkit research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 36. ACM, 2018.

[LKB05] Gun A Lee, Gerard J Kim, and Mark Billinghurst. Immersive authoring: What you experience is what you get (wyxiwyg). *Communications of the ACM*, 48(7):76–81, 2005.

[LMZ+12] Tobias Langlotz, Stefan Mooslechner, Stefanie Zollmann, Claus Degendorfer, Gerhard Reitmayr, and Dieter Schmalstieg. Sketching up the world: in situ authoring for mobile augmented reality. *Personal and ubiquitous computing*, 16(6):623–630, 2012.

[LNBK04] Gun A Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 172–181. IEEE, 2004.

[LOG17] David Ledo, Lora Oehlberg, and Saul Greenberg. The toolkit-audience challenge. In *Workshop on HCI. Tools at CHI 2017*, 2017.

[LPKW06] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-user development: An emerging paradigm. In *End user development*, pages 1–8. Springer, 2006.

[LS05] Florian Ledermann and Dieter Schmalstieg. *APRIL: a high-level framework for creating augmented reality presentations*. IEEE, 2005.

[LS17] James R Lewis and Jeff Sauro. Can i leave this one out?: the effect of dropping an item from the sus. *Journal of Usability Studies*, 13(1):38–46, 2017.

[LTK+07] Michael Leitner, Martin Tomitsch, Thomas Költringer, Karin Kappel, and Thomas Grechenig. Designing tangible table-top interfaces for patients in rehabilitation. In *CVHI*, 2007.

[Mac02] Blair MacIntyre. Authoring 3d mixed reality experiences: Managing the relationship between the physical and virtual worlds. *At ACM SIGGRAPH and Eurographics Campfire: Production Process of 3D Computer Graphics Applications-Structures, Roles and Tools, Snowbird, UT*, pages 1–5, 2002.

[Mac05] Asa MacWilliams. *A decentralized adaptive architecture for ubiquitous augmented reality systems*. PhD thesis, Technische Universität München, 2005.

[Mar07] Paul Marshall. Do tangible interfaces enhance learning? In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 163–170, 2007.

[May02] Richard E Mayer. Multimedia learning. In *Psychology of learning and motivation*, volume 41, pages 85–139. Elsevier, 2002.

[MBC13] Javier Marco, Sandra Baldassarri, and Eva Cerezo. Nikvision: Developing a tangible application for and with children. *J. UCS*, 19(15):2266–2291, 2013.

[MCB12] Javier Marco, Eva Cerezo, and Sandra Baldassarri. Toyvision: a toolkit for prototyping tabletop tangible games. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 71–80, 2012.

[ME19] Tariq Masood and Johannes Egger. Adopting augmented reality in the age of industrial digitalisation. 2019.

[MGB$^+$03] Blair MacIntyre, Maribeth Gandy, Jay Bolter, Steven Dow, and Brendan Hannigan. Dart: The designer's augmented reality toolkit. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 329–330. IEEE, 2003.

[MHP00] Brad Myers, Scott E Hudson, and Randy Pausch. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):3–28, 2000.

[MK94] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[MKRL86] Desire L Massart, Leonard Kaufman, Peter J Rousseeuw, and Annick Leroy. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, 187:171–179, 1986.

[MM09] Pranav Mistry and Pattie Maes. Sixthsense: a wearable gestural interface. In *ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation*, pages 85–85. ACM, 2009.

[MO12] Andrew Manches and Claire O'malley. Tangibles for learning: a representational analysis of physical manipulation. *Personal and Ubiquitous Computing*, 16(4):405–419, 2012.

[MPW08] Valérie Maquil, Thomas Psik, and Ina Wagner. The colortable: a design story. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 97–104, 2008.

[MSW$^+$03] Asa MacWilliams, Christian Sandor, Martin Wagner, Martin Bauer, Gudrun Klinker, and Bernd Bruegge. Herding sheep: live system for distributed augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 123–132. IEEE, 2003.

[MTA+17] Valerie Maquil, Eric Tobias, Dimitra Anastasiou, Hélène Mayer, and Thibaud Latour. Copse: Rapidly instantiating problem solving activities based on tangible tabletop interfaces. *Proceedings of the ACM on Human-Computer Interaction*, 1(EICS):1–16, 2017.

[Muñ17] Hendrys Fabián Tobar Muñoz. *Supporting technology for augmented reality game-based learning.* PhD thesis, Universitat de Girona, 2017.

[MWS12] Milena S Markova, Stephanie Wilson, and Simone Stumpf. Tangible user interfaces for learning. *International Journal of Technology Enhanced Learning*, 4(3-4):139–155, 2012.

[Nie94] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[NM90] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, 1990.

[Nor88] Donald A Norman. *The psychology of everyday things.* Basic books, 1988.

[NPD13] Lene Nielsen, M Soegaard Personas, and RF Dam. The encyclopedia of human-computer interaction. *The Interaction Design Foundation. Aarhus, Denmark. Available at: http://www. interaction-design. org/encyclopedia/personas. html*, 2013.

[NRD15] Rafael Nunes, Fabio Rito, and Carlos Duarte. Tactic: an api for touch and tangible interaction. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 125–132, 2015.

[OF04] Claire O'Malley and Danae Stanton Fraser. Literature review in learning with tangible technologies. *Bristol: NESTA Futurelab*, 2004.

[OJ07] Dan R Olsen Jr. Evaluating user interface systems research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 251–258. ACM, 2007.

[Ols92] Dan Olsen. *User interface management systems: models and algorithms*. Morgan Kaufmann Publishers Inc., 1992.

[PD95] Robert W Proctor and Addie Dutta. *Skill acquisition and human performance.* Sage Publications, Inc, 1995.

[Pet77] James L Peterson. Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252, 1977.

[PFSR09] Sara Price, Taciana Pontual Falcão, Jennifer G Sheridan, and George Roussos. The effect of representation location on interaction in a tangible learning environment. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 85–92, 2009.

[Poe06] Iman Poernomo. The meta-object facility typed. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1845–1849, 2006.

[PROM18] Maria Madalena Paulo, Paulo Rita, Tiago Oliveira, and Sérgio Moro. Understanding mobile augmented reality adoption in a consumer context. *Journal of hospitality and tourism technology*, 2018.

[PTB$^+$01] Ivan Poupyrev, Desney S Tan, Mark Billinghurst, Hirokazu Kato, Holger Regenbrecht, and Nobuji Tetsutani. Tiles: A mixed reality authoring interface. In *Interact*, volume 1, pages 334–341, 2001.

[PTTVG03] Fred Paas, Juhani E Tuovinen, Huib Tabbers, and Pascal WM Van Gerven. Cognitive load measurement as a means to advance cognitive load theory. *Educational psychologist*, 38(1):63–71, 2003.

[PW09] Jonghee Park and Woontack Woo. Multi-layer based authoring tool for digilog book. In *International Conference on Entertainment Computing*, pages 234–239. Springer, 2009.

[RA09] Philip Robbins and Murat Aydede. A short primer on situated cognition. *The Cambridge handbook of situated cognition*, pages 3–10, 2009.

[RÁMGHH16] David Roldán-Álvarez, Estefanía Martín, Manuel García-Herranz, and Pablo A Haya. Mind the gap: Impact on learnability of user interface design of authoring tools for teachers. *International Journal of Human-Computer Studies*, 94:18–34, 2016.

[RÁMH$^+$18] David Roldán-Álvarez, Estefanía Martín, Pablo A Haya, Manuel García-Herranz, and María Rodríguez-González. Dedos: An authoring toolkit to create educational multimedia activities for multiple devices. *IEEE Transactions on Learning Technologies*, 11(4):493–505, 2018.

[RCES09] Abu Saleh Md Mahfujur Rahman, Jongeun Cha, and Abdulmotaleb El Saddik. Authoring edutainment content through video annotations and 3d model augmentation. In *2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurements Systems*, pages 370–374. IEEE, 2009.

[RdFST13] Rafael Alves Roberto, Daniel Queiroz de Freitas, Francisco Paulo Magalhaes Simões, and Veronica Teichrieb. A dynamic blocks platform based on projective augmented reality and tangible interfaces for educational activities. In *2013 XV Symposium on Virtual and Augmented Reality*, pages 1–9. IEEE, 2013.

[RF17] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[RHM$^+$13] Stefan Rüther, Thomas Hermann, Maik Mracek, Stefan Kopp, and Jochen Steil. An assistance system for guiding workers in central sterilization supply departments. In *Proceedings of the 6th International Conference* *on PErvasive Technologies Related to Assistive Environments*, pages 1–8, 2013.

[RLMT16] Rafael Alves Roberto, João Paulo Lima, Roberta Cabral Mota, and Veronica Teichrieb. Authoring tools for augmented reality: An analysis and classification of content design tools. In *International Conference of Design, User Experience, and Usability*, pages 237–248. Springer, 2016.

[RM09] Iulian Radu and Blair MacIntyre. Augmented-reality scratch: a tangible programming environment for children. In *Proceedings of conference on interaction design for children, Como, Italy*, 2009.

[RMMH+09] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[Rog03] Everett M. Rogers. *Diffusion of innovations*. Free Press, 2003.

[RRMSMC18] Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76:38–47, 2018.

[RS99] Jun Rekimoto and Masanori Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 378–385, 1999.

[RT07] Peter Robinson and Philip Tuddenham. Distributed tabletops: Supporting remote and mixed-presence tabletop collaboration. In *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*, pages 19–26. IEEE, 2007.

[RVBB+06] Ramesh Raskar, Jeroen Van Baar, Paul Beardsley, Thomas Willwacher, Srinivas Rao, and Clifton Forlines. ilamps: geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2006 Courses*, pages 7–es. ACM, 2006.

[RWLB01] Ramesh Raskar, Greg Welch, Kok-Lim Low, and Deepak Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Rendering Techniques 2001*, pages 89–102. Springer, 2001.

[San05] Christian Sandor. *A software toolkit and authoring tools for user interfaces in ubiquitous augmented reality*. PhD thesis, Technische Universität München, 2005.

[San15] Marc Ericson C. Santos. *Usability Evaluation Framework for Handheld Augmented Reality Applied to Learning Support*. PhD thesis, Nara Institute of Science and Technology, 2015.

[Sev12] Charles Severance. Discovering javascript object notation. *Computer*, 45(4):6–8, 2012.

[SFH$^+$02] Dieter Schmalstieg, Anton Fuhrmann, Gerd Hesina, Zsolt Szalavári, L Miguel Encarnaçao, Michael Gervautz, and Werner Purgathofer. The studierstube augmented reality project. *Presence: Teleoperators & Virtual Environments*, 11(1):33–54, 2002.

[SH$^+$10] Orit Shaer, Eva Hornecker, et al. Tangible user interfaces: past, present, and future directions. *Foundations and Trends® in Human–Computer Interaction*, 3(1–2):4–137, 2010.

[SH16] Dieter Schmalstieg and Tobias Hollerer. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[SI11] Luca Simeone and Salvatore Iaconesi. Anthropological conversations: Augmented reality enhanced artifacts to foster education in cultural anthropology. In *2011 IEEE 11th International Conference on Advanced Learning Technologies*, pages 126–128. IEEE, 2011.

[SJZD10] Bertrand Schneider, Patrick Jermann, Guillaume Zufferey, and Pierre Dillenbourg. Benefits of a tangible interface for collaborative learning and interaction. *IEEE Transactions on Learning Technologies*, 4(3):222–232, 2010.

[SK05] Christian Sandor and Gudrun Klinker. A rapid prototyping software infrastructure for user interfaces in ubiquitous augmented reality. *Personal and Ubiquitous Computing*, 9(3):169–185, 2005.

[SKY$^+$14] Jinwook Shim, Minje Kong, Yoonsik Yang, Jonghoon Seo, and Tack-Don Han. Interactive features based augmented reality authoring tool. In *2014 IEEE International Conference on Consumer Electronics (ICCE)*, pages 47–50. IEEE, 2014.

[SLB08] Hartmut Seichter, Julian Looser, and Mark Billinghurst. Composar: An intuitive tool for authoring ar applications. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 177–178. IEEE, 2008.

[SLB11] Dieter Schmalstieg, Tobias Langlotz, and Mark Billinghurst. Augmented reality 2.0. In *Virtual realities*, pages 13–37. Springer, 2011.

[SLR$^+$14] Marc Ericson C SANTOS, Arno In Wolde LUEBKE, Ma Mercedes T RODRIGO, Takafumi TAKETOMI, Goshiro YAMAMOTO, Christian Sandor, KATO Hirokazu, et al. Authoring augmented reality as situated multimedia. In *22nd International Conference on Computers in Education (ICCE 2014)*, pages 554–556. Asia-Pacific Society for Computers in Education, 2014.

[SMF16] Genta Suzuki, Taichi Murase, and Yusaku Fujii. Projecting recorded expert hands at real size, at real speed, and onto real objects for manual work. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces*, pages 13–17. ACM, 2016.

[SMMM17] Lou Schwartz, Valérie Maquil, Christian Moll, and Hélène Mayer. Teachers' feedback on the end-user development of tangible systems. *Mensch und Computer 2017-Workshopband*, 2017.

[SPC$^+$16] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016.

[SSDM17] Ivo Sluganovic, Matej Serbec, Ante Derek, and Ivan Martinovic. Holopair: Securing shared augmented reality using microsoft hololens. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 250–261. ACM, 2017.

[SSFG98] Zsolt Szalavári, Dieter Schmalstieg, Anton Fuhrmann, and Michael Gervautz. "studierstube": An environment for collaboration in augmented reality. *Virtual Reality*, 3(1):37–48, 1998.

[STY$^+$14] Marc Ericson C SANTOS, Takafumi TAKETOMI, Goshiro YAMAMOTO, Ma Mercedes T RODRIGO, Christian SANDOR, KATO Hirokazu, et al. Evaluating augmented reality for situated vocabulary learning. In *22nd International Conference on Computers in Education (ICCE 2014)*, pages 701–710. Asia-Pacific Society for Computers in Education, 2014.

[SWBP13] Bertrand Schneider, Jenelle Wallace, Paulo Blikstein, and Roy Pea. Preparing for future learning with a tangible user interface: the case of neuroscience. *IEEE Transactions on Learning Technologies*, 6(2):117–129, 2013.

[SWMJ10] Steven C Seow, Dennis Wixon, Ann Morrison, and Giulio Jacucci. Natural user interfaces: the prospect and challenge of touch and gestural computing. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 4453–4456. ACM, 2010.

[TBB10] Tiffany Tseng, Coram Bryant, and Paulo Blikstein. Mechanix: an interactive display for exploring engineering design through a tangible interface. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, pages 265–266, 2010.

[TJR$^+$19] Lauren Thevin, Christophe Jouffrais, Nicolas Rodier, Nicolas Palard, Martin Hachet, and Anke M Brock. Creating accessible interactive audio-tactile drawings using spatial augmented reality. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*, pages 17–28, 2019.

[TML15] Eric Tobias, Valérie Maquil, and Thibaud Latour. Tulip: a widget-based software framework for tangible tabletop interfaces. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 216–221, 2015.

[TSM13] Daniel Tetteroo, Iris Soute, and Panos Markopoulos. Five key challenges in end-user development for tangible and embodied interaction. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 247–254, 2013.

[TVG+15] Daniel Tetteroo, Paul Vreugdenhil, Ivor Grisel, Marc Michielsen, Els Kuppens, Diana Vanmulken, and Panos Markopoulos. Lessons learnt from deploying an end-user development platform for physical rehabilitation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4133–4142. ACM, 2015.

[UFG+16] Antonio Emmanuele Uva, Michele Fiorentino, Michele Gattullo, Marco Colaprico, Maria F de Ruvo, Francescomaria Marino, Gianpaolo F Trotta, Vito M Manghisi, Antonio Boccaccio, Vitoantonio Bevilacqua, et al. Design of a projective ar workbench for manual working stations. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pages 358–367. Springer, 2016.

[UI97] Brygg Ullmer and Hiroshi Ishii. The metadesk: models and prototypes for tangible user interfaces. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 223–232, 1997.

[UI98] John Underkoffler and Hiroshi Ishii. Illuminating light: an optical design tool with a luminous-tangible interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 542–549, 1998.

[UI99] John Underkoffler and Hiroshi Ishii. Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 386–393, 1999.

[UI00] Brygg Ullmer and Hiroshi Ishii. Emerging frameworks for tangible user interfaces. *IBM systems journal*, 39(3.4):915–931, 2000.

[UIJ05] Brygg Ullmer, Hiroshi Ishii, and Robert JK Jacob. Token+ constraint systems for tangible interaction with digital information. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(1):81–118, 2005.

[UNT+12] Daisuke Uriu, Mizuki Namai, Satoru Tokuhisa, Ryo Kashiwagi, Masahiko Inami, and Naohito Okude. Panavi: recipe medium with a sensors-embedded pan for domestic users to master professional culinary arts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 129–138, 2012.

[UUI99] John Underkoffler, Brygg Ullmer, and Hiroshi Ishii. Emancipated pixels: real-world graphics in the luminous room. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 385–392, 1999.

[VD97] Andries Van Dam. Post-wimp user interfaces. *Communications of the ACM*, 40(2):63–67, 1997.

[VH01] Eric Von Hippel. User toolkits for innovation. *Journal of Product Innovation Management: An International Publication Of The Product Development & Management Association*, 18(4):247–257, 2001.

[WAF19] Daniel Wessel, Christiane Attig, and Thomas Franke. Ati-s-an ultra-short scale for assessing affinity for technology interaction in user studies. In *Proceedings of Mensch und Computer 2019*, pages 147–154. ACM, 2019.

[WC13] Rafał Wojciechowski and Wojciech Cellary. Evaluation of learners' attitude toward learning in aries augmented reality environments. *Computers & Education*, 68:570–585, 2013.

[Wel91] Pierre Wellner. The digitaldesk calculator: tangible manipulation on a desk top display. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 27–33, 1991.

[WF09] Sean White and Steven Feiner. Sitelens: situated visualization techniques for urban site visits. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1117–1120, 2009.

[Woj12] Rafał Wojciechowski. Modeling interactive augmented reality environments. In *Interactive 3D Multimedia Content*, pages 137–170. Springer, 2012.

[WON16] Xiangyu Wang, Soh K Ong, and Andrew YC Nee. A comprehensive survey of augmented reality assembly research. *Advances in Manufacturing*, 4(1):1–22, 2016.

[WQ10] Rui Wang and Xuelei Qian. *OpenSceneGraph 3.0: Beginner's guide*. Packt Publishing Ltd, 2010.

[WTS10] Ming-Jen Wang, Chien-Hao Tseng, and Cherng-Yeu Shen. An easy to use augmented reality authoring tool for use in examination purpose. In *IFIP Human-Computer Interaction Symposium*, pages 285–288. Springer, 2010.

[WW11] Daniel Wigdor and Dennis Wixon. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.

[WZLL13] JF Wang, C Zeng, Y Liu, and SQ Li. Integrated content authoring for augmented reality based product manual assembly process instruction. In *The 43rd International Conference on Computers and Industrial Engineering*, pages 242–251, 2013.

[ZAR05] Oren Zuckerman, Saeed Arida, and Mitchel Resnick. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 859–868, 2005.

[ZGN$^+$11] Ryder Ziola, Shweta Grampurohit, L Nate, James Fogarty, and Beverly Harrison. Oasis: Creating smart objects with dynamic digital behavior. In *Workshop at IUI*, volume 2011, 2011.

[ZH04] Jürgen Zauner and Michael Haller. Authoring of mixed reality applications including multi-marker calibration for mobile devices. In *Proceedings of the Tenth Eurographics conference on Virtual Environments*, pages 87–90, 2004.

[ZHBH03] Jürgen Zauner, Michael Haller, Alexander Brandl, and Werner Hartman. 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.*, pages 237–246. IEEE, 2003.

[ZJLD09] Guillaume Zufferey, Patrick Jermann, Aurélien Lucchi, and Pierre Dillenbourg. Tinkersheets: using paper forms to control and visualize tangible simulations. In *Proceedings of the 3rd international Conference on Tangible and Embedded interaction*, pages 377–384, 2009.

[ZON13] J Zhu, SK Ong, and AYC Nee. An authorable context-aware augmented reality system to assist the maintenance technicians. *The International Journal of Advanced Manufacturing Technology*, 66(9-12):1699–1714, 2013.

[ZSDS13] Emile Zhang, Hideo Saito, and Francois De Sorbier. From smartphone to virtual window. In *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2013.