Story ARtist

Story Authoring in Augmented Reality

by

Marie Kegeleers

to obtain the degree of Master of Science
at the Delft University of Technology.

Thesis committee:	Dr. ir. Rafael Bidarra, supervisor	TU Delft - Computer Graphics and Visualization
	Dr. ir. Ricardo Marroquim	TU Delft - Computer Graphics and Visualization
	Dr. Myrthe Tielman	TU Delft - Interactive Intelligence

Preface

Throughout my years at the TU Delft, I had many incredible opportunities to work on interesting projects using fascinating technology, be part of communities filled with enthusiastic people with similar interests, gain valuable work experience in my field of interest, computer science, meet and work with incredible people, from fellow students to professors and colleagues, and much more. One of those opportunities happened during the first year of my master's degree, when I had the opportunity to work on an augmented reality (AR) project with the Microsoft Hololens for a course. This was one of the most intense but also one of the best courses of my student career. Such an AR project was rewarding to work on and the result was an achievement for the whole team. A few months later, I was offered to do my master thesis on the subject of augmented reality. Of course, this was an offer I could not refuse. The topic we settled on was story authoring. After navigating the topic for a while, some prototypes were made, interaction techniques were explored and finally, I developed the Story ARtist application. The most rewarding part, however, was watching people interact with the system I built and witnessing their reactions, especially when they enjoyed their experience and understood the purpose and potential of this AR application.

Augmented reality is an amazing kind of technology in which I see great potential for the future in many different application fields. I can only hope this thesis helps you see a part of this potential too.

This project would not have been possible without the help and support of certain people, who I would like to thank. I would like to begin by thanking Ruud de Jong for providing me with all the hardware I needed for my ever-growing setup. I would like to thank Mijael Bueno for sharing his knowledge about story representation and taking the time to discuss it with me. I also want to thank Tom Belet for providing me with 3D models which made the AR application built for this project look a lot more appealing. I want to thank my thesis committee, Myrthe Tielman, with a special thanks for providing me with great advice about evaluating interactive systems, and Ricardo Marroquim, for evaluating this thesis work. I would like to thank all user study participants for taking the time to test my application and providing a lot of valuable feedback. I also would like to thank my family, boyfriend and friends for their continued support and for providing me with some distraction from my work, whenever it was needed.

Finally, I would like to thank my supervisor, Rafael Bidarra, for his guidance, enthusiasm and encouragement to go the extra mile. I am very grateful to have had him as my supervisor, not only for this thesis, but throughout my years at the TU Delft.

Marie Kegeleers
Delft, August 2020

Contents

1 Introduction 1

2 Related Work 3
2.1 AR Interfaces 3
2.2 AR Content Creation 6
2.3 Story Authoring Tools. 7

3 AR Interaction Design 9
3.1 Prototypes 9
3.2 Combining Tangible and Touch-less 13

4 Story ARtist Application 15
4.1 Plot Point Structure 15
4.2 Interface. 16
4.3 Example Story 19

5 Implementation 21
5.1 Setup 21
5.2 Story Framework 23
5.3 Software Architecture 24

6 Evaluation 27
6.1 User Study 27
6.2 Data Processing 28
6.3 Results 29
6.4 Discussion 32

7 Conclusion 35

A Resource Files 37

B Evaluation Materials 41
B.1 Interview Questions 41
B.2 Questionnaire. 41

Bibliography 43

1 Introduction

Content creation applications that are commonly used today are generally regular PC applications with visualisation on a 2D display and interaction using a keyboard and mouse. This includes applications for 3D content creation, e.g. 3D modeling, where a 3D view is simulated on the 2D display. Other options for mediums and interfaces might be worth considering as they could be more suitable for this type of applications. Furthermore, using less optimal and complicated interfaces and visualisation can make 3D content creation less accessible since some experience would be needed to fluently control the application. The Graphical User Interface (GUI) has been the standard for years. However, a new kind of user interface gradually appears in more research and applications, the Natural User Interface (NUI). A NUI is an interface that enables users to interact with virtual elements in a way similar to how people interact with the real world. It makes use of everyday actions like gestures, touching and picking up objects, and speech for the user to control the application [1]. Because of this, a NUI can have a more direct way of interacting with virtual content which, as the name suggests, can be perceived as more natural compared to a GUI.

A medium that enables the use of a NUI is Augmented Reality (AR). AR is the technology that adds virtual elements to a real-world environment. It provides a digital overlay on top of reality, superimposing digital information on the world we see. Because of this integration with reality, AR provides a very different way of interacting with virtual content. The interface is not limited to a 2D screen but can occupy a 3D space around the user. The same goes for visualisation of virtual content which can be displayed in front of the user as a 3D hologram. This is a more representative visualization than a flat image on a 2D screen. Another advantage is that it enables easy exploration through physical movement, for example viewing the scene from different angles or examining objects up close.

AR is a promising new platform that is not commonly used yet while it could be helpful in many different fields. Any field that benefits from real 3D visualisations, not shown on a 2D screen, and a more hands-on and natural interaction, could be improved using AR. One of these fields that could benefit from AR is content creation. Although researchers and developers are examining and exploring how AR can be utilized as a helpful tool in a wide range of applications, it is currently not common to research AR interfaces for creative tools and even more rare to actually use them.

Because the user's environment and body are visible, creative AR tools allow creations to be shown in a real-world context and modified using natural operations, either using tangibles or movement of body parts. This does not require the user to learn new unfamiliar actions to perform the right operations on the virtual content. Instead, a NUI can be used that exploits actions that the user is already familiar with.

In this thesis, the type of content creation that will be focused on is story authoring. This was chosen as a proof of concept for this project because story authoring is an accessible way of creating content, as anyone can write a story by composing simple elements like characters, objects and actions. Furthermore, story authoring maintains enough similarities in its functionality with other types of creative tools. In the past years, research has been done on ways to facilitate story authoring using technology and some story authoring tools have been developed for desktop PC environments.

However, because of its 3D visualisation and the use of a NUI, AR could be used to improve these applications and make them more accessible. To verify this, this research aims to answer the following question: How can AR be used for story authoring? The goal of this project is to evaluate the use of AR for story authoring with a focus on accessibility, and identify the benefits. To do this, both types of existing AR interfaces, tangible and touch-less, will be researched with the goal of including them in a single NUI to find fitting combinations for an AR interface for story authoring. More specifically, interaction with markers and hand tracking will be explored and combined to best accommodate all kinds of interactions. A prototype application, Story ARtist, will be developed to evaluate the possibilities for AR interaction and visualisation for story authoring. Story ARtist will introduce new interaction and visualisation concepts that aim to make story authoring, and possibly content creation in general, accessible. Because of the spatiality of AR, the Story ARtist interface can occupy the entire 3D space. To represent the story in the application, a story framework will be developed that is easy to understand and focuses on the core elements of a story: actions, characters, objects and environments. Because of the use of AR, story elements can be placed in a 3D scene that occupies a physical 3D space which can help the author visualise their authored scene.

The NUI will require the author to use their hands to interact with markers or virtual elements directly. Using a hand-held AR device, like a phone or tablet, would cause the author to have only one hand available. Therefore, to enable optimal hand interaction, a head-mounted AR device will be used. To have all elements within reach, the application will be developed for a tabletop environment, as this is a convenient workspace for content creation applications.

2 Related Work

To be able to evaluate AR for story authoring, an AR concept application will be needed. For the user to be able to interact with virtual elements, an interface will be needed, which is an important focus of the application. Therefore, existing AR interfaces for a variety of applications will be discussed. Some examples of research on content creation applications in AR will be reviewed as well. For the story authoring side of the project, some existing technological story authoring applications will be described.

2.1. AR Interfaces

Existing types of AR interfaces can be classified into two categories. The first category is tangible interfaces, discussed in Section 2.1.1. These interfaces require the user to interact with physical elements like blocks or cards to control virtual elements. The second category is touch-less interfaces, described in Section 2.1.2. These interfaces are fully virtual and do not involve any physical elements. The user interacts only with virtual elements like menu panels with buttons or holograms using gestures and hand interaction, for example. Both types of interfaces provide very different experiences and fit different applications based on the AR equipment used, the environment and the application purpose.

2.1.1. Tangible Interfaces

Tangible interfaces combine the overlaid virtual elements with physical objects. The user can manipulate and interact with these physical objects to control virtual elements. The power of tangible interfaces is that the interactive elements have physical properties and constraints that the user is familiar with. This restricts how the objects can be manipulated and therefore makes the controls easy and intuitive [2, 3].

Types of Tangible Interfaces

The most common implementation of tangible AR interfaces is cards with markers. The most basic interface consists of regular cards with some design printed on them, that each represent a single virtual element in AR. Kato et al. [2] designed an interface like this for a collaborative and interactive AR application where cards had to be matched based on their AR content. When a user brings a card into view, the corresponding AR object is displayed, making it visible for the user when looking through the AR device, as can be seen in Figure 2.1. When the card is moved or rotated, the virtual object follows. This allows the user to look at the virtual object from all angles. The interface also includes interaction between virtual objects. When matching cards are held close together, the corresponding virtual elements will start interacting.

To allow modifications of and interactions with virtual elements, markers do not necessarily all need to correspond to a virtual element but can also represent an action. Poupyrev et al. [4] developed a tangible AR interface where marker cards were divided into data tiles, each containing a virtual element, and operation tiles, representing an action. Moving an operation tile next to a data tile causes the operation to be performed on the data tile.

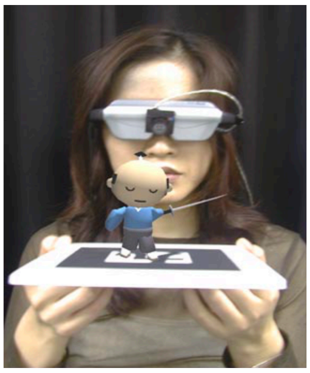


Figure 2.1: A card containing a marker with a virtual object displayed on top [2].

Another possibility is to have certain markers represent virtual objects and have other markers represent properties of those objects such that when a property card comes into view, the corresponding property of the virtual element changes. An example application for this is interior design [5], where a set of markers represents different types of furniture and interior objects, and another set of markers represents a property, for example, colour. Moving the furniture marker causes the virtual furniture to move, changing the property marker will change the colour of the furniture. Another idea introduced by Poupyrev et al. [4] is dynamically assigned markers. In most applications, each marker card has a predetermined virtual element assigned to it. However, to improve flexibility and reduce the number of marker cards needed, it is possible to make the user assign objects to cards dynamically. By providing one element that represents a virtual catalog and designing actions to copy objects from the catalog to marker cards, the user can select the object they need and assign those to empty markers.

This type of interface is often used on a surface like a table (horizontal) or a whiteboard (vertical). Because of this, a possible extension is to not anchor the virtual elements to their card but instead use one marker placed on the surface as the base for all virtual elements and spawn the elements on this surface when a new marker card is introduced. An example of an application with this mechanic is the interactive narrative authoring tool developed by Kapadia et al. [6]. One specific large marker defines the narrative's environment. The characters freely walk around on the surface where the environment marker is placed. When other markers are in view, a corresponding object is spawned or an action is triggered somewhere in the environment, not specifically on the marker. This way, the virtual elements can interact more freely but the user no longer has direct control over them. This type of interface can be beneficial for applications where virtual elements have their own animations and logic, like the application by Kapadia et al. where the main focus is on influencing the narrative and how the characters behave and interact.

By allowing the markers to represent more complex types of objects with more complicated interactions, and by using the markers on different kinds of objects, other than cards, more sophisticated interfaces can be created. For example, one could make a specific marker represent an interaction cursor, like a virtual pen, as displayed in Figure 2.2. The pen can be used to interact with virtual panels and buttons or perform operations on virtual content. An example application for this is product design and modeling [7]. The pen can be used to select actions on a virtual button panel and to interact with entities of the object being modeled. This way, the user can make modifications in real-time and view their changes in the environment from all angles.

Limitations

There are also some disadvantages to tangible interfaces. Physical objects have their natural properties which can be difficult to change and therefore very limiting [2]. Physical properties make tangible elements and their controls easy to use. However, they restrict the possibilities of AR. Furthermore, markers and other tangible objects always need a decent size for optimal tracking. This can result in a less convenient way of interacting and can be limiting for precise interactions.

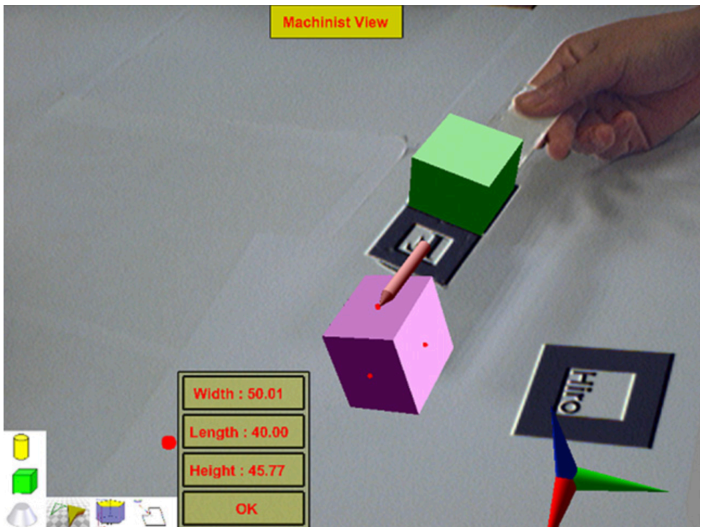


Figure 2.2: A marker representing a virtual pen that can be used to interact with virtual elements [7].

2.1.2. Touch-less Interfaces

An alternative to tangible interfaces that could overcome some of the limitations of physical objects is touch-less interfaces. As the name suggests, touch-less interfaces are fully virtual and therefore do not require the user to physically touch anything.

Types of Touch-less Interfaces

A common way to interact with a touch-less interface is through body gestures, more specifically hand gestures. A simple and intuitive way to use gestures is to use one or two fingers as a pointer, similar to a mouse in a regular desktop setup. The user can move this virtual mouse by moving their hand, so the position of the fingers in the environment can be used as input. An example of fingers being used as a virtual mouse can be seen in Figure 2.3, which shows an AR football game where the player controls a goalkeeper's glove by moving their fingers. This game was developed by Lv et al. [8] who developed a finger and foot tracking method for AR applications. If virtual buttons are used, a selection command can be simulated by bending and extending the fingers, similar to a clicking motion when using a regular mouse. Another option for selection is to detect a clicking action when the fingertip stays on a button for a certain amount of time [9]. When more complex gestures than pointer motions can be recognized by the system, more possibilities open up for intuitive interaction. Benko et al. [10] presented a collaborative mixed reality tool for archaeology that includes a hand tracking glove that allows the user to grab a virtual object. Reaching for an object with an open hand followed by closing their hand inside or near the object, i.e. a grabbing motion, attaches the object to their hand to be able to move it around to examine.

Another type of AR interaction using hand tracking is through non-direct hand gestures. Instead of directly manipulating a pointer or other virtual elements, predefined hand gestures performed separately from virtual elements trigger corresponding actions. This was demonstrated by Ng et al. [11]. For example, having a virtual model in view, a thumbs-up gesture corresponds to rotation around the model's y-axis. To emulate a mouse, the gesture to make the mouse move to the left could be pointing to the left, while keeping the hand stationary. Non-direct gestures can be helpful in applications where it is difficult to modify virtual elements on a large scale, larger than the user's arm length, or when it is difficult for the user to move around.

Limitations

Although hand gesture recognition is considered one of the most natural ways to interact in AR [9, 12], not many research papers are available about this topic compared to tangible interfaces. Most touch-less interfaces require more advanced software and/or hardware. To track the user's hands, either advanced hand tracking software based on computer vision is needed, a specialized tracking glove is required or a special type of camera needs to be used. Recent developments in computer vision based

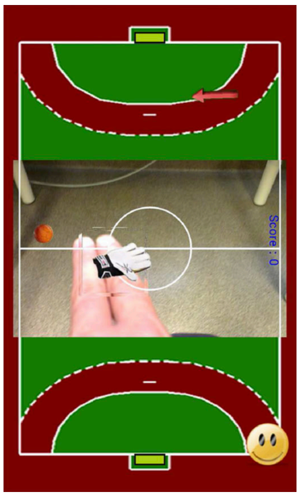


Figure 2.3: Two fingers being used as input for a pointer [8].

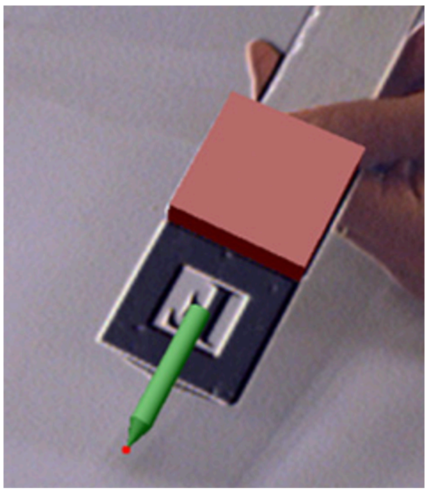


Figure 2.4: A marker-based interaction stylus [7].

hand tracking [13, 14] and the implementation of this feature in AR devices, make hand gesture based interfaces very promising for the future. However, because these techniques are relatively new and not fully optimised, easily accessible and/or feasible yet, hand tracking is not perfectly robust.

Many researchers in the past years, especially during the first years of AR research, chose the simplest and easiest type of interface to implement, which is a tangible interface using markers. Simple image processing techniques using a regular camera suffice to track the position and orientation of these markers so no advanced tracking techniques or hardware are required. As mentioned above, hand tracking and gesture recognition are advancing which means that touch-less interfaces may become more common in the future. Because of these developments, we might currently be at a turning point where a majority of the future research on AR might be focused on or, at least, involve touch-less interfaces using hand gestures. If research shows more promising results when AR systems employ touch-less interfaces, which is possible because touch-less interfaces overcome some of the limitations of tangible interfaces, the use of tangible interfaces might decline significantly.
However, markers and other tangible interfaces have their advantages and could counteract the limitations of touch-less interfaces. Their simplicity does not prevent tangible interfaces from being a powerful interaction tool. By combining touch-less with tangible, where each type of interface is used for the interactions that suit them best, a more optimal interface could be created.

2.2. AR Content Creation

Content creation applications for 3D content can benefit from the 3D aspects of AR. Most commonly, these applications are regular PC applications with visualisation on a 2D screen and a 2D interface. Some research has been done on content creation in AR where new interface concepts were introduced. Shen et al. [7] created an AR product design application that includes 3D modeling and collaborative design activities. The main interaction tool for modeling is a virtual stylus that is controlled by two markers placed next to each other. The first marker is used for position tracking, the second marker is used as a selection mechanic. When the second marker is occluded, it registers as a button click, selecting what is currently at the tip of the stylus. The stylus can be seen in Figure 2.4. Phan and Choo [5] designed an AR application for interior design with a similar interaction mechanic using markers and occlusion. Furniture can be arranged in a room using single markers representing a piece of furniture to track the position. Strips of markers placed next to each other can be used to change the properties of the furniture. The property changes based on which marker is occluded. For example, placing a strip with markers representing colours next to the positional marker of a chair, allows the user to change the colour of the chair by occluding the marker corresponding to the desired colour. This can be seen in Figure 2.5.

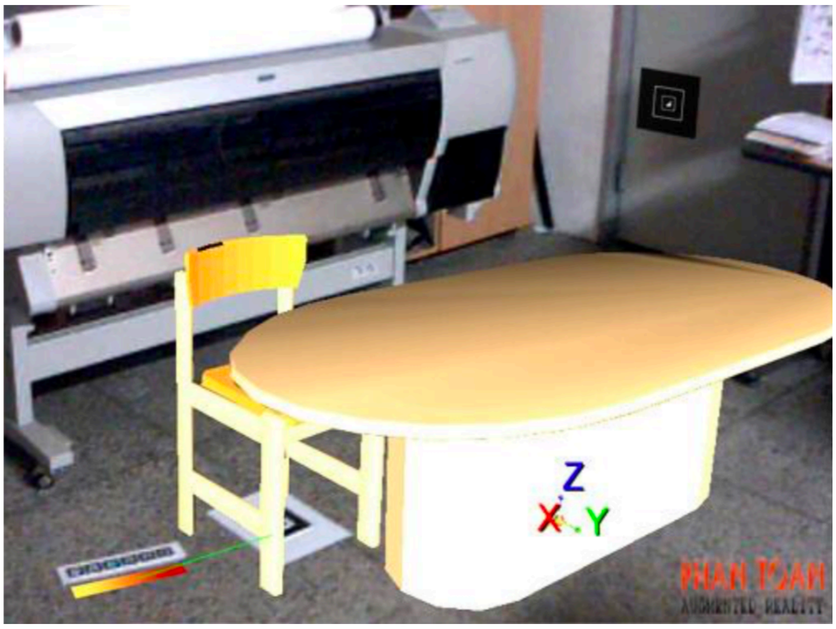


Figure 2.5: Example of a strip of markers representing a range of colours (bottom left) [5].

Notably, a lot of research on content creation in AR uses tangible interfaces, markers to be more specific. Not much research has been done on AR content creation using touch-less interfaces, even though touch-less interaction could be a better fit for this type of application. For example, hand interaction for 3D modeling allows the user to manipulate the content with their hands similar to manipulating a physical model. This is more intuitive than, for example, positioning and occluding markers to manipulate the virtual content. A possible explanation for the lack of AR content creation research using touch-less interfaces is the accessibility and simplicity of marker tracking. It only requires some pieces of paper with a design printed on them, a basic camera and an image processing algorithm that is relatively simple.

As discussed above, touch-less interfaces typically use more advanced techniques like hand tracking. With marker-less tracking techniques becoming more advanced and more common, as they are being integrated into AR devices, new interface tools become more accessible. This allows for further research on AR content creation using touch-less mechanics instead of only tangible marker-based interfaces.

2.3. Story Authoring Tools

There are many different ways a person can author a story. From traditional writing to making a film or video or making choices in a video game with a variable storyline. In the past years, research has been done on ways to facilitate story authoring using technology to increase accessibility. Some story authoring tools have been developed for desktop environments.

An example of an interactive story building framework is StoryTec by Göbel et al. [15]. The framework provides everything to create a digital interactive story, all combined into one GUI. It consists of separate editors for the overall story, the scene, the actions and more. All these editors are displayed in separate windows with a different representation of (a part of) the story. In the story editor, for example, the story is represented as a structured graph. The nodes correspond to scenes or grouped scenes and edges show transitions between the scenes. This provides the user with a clean and scalable visualization of the entire story. Another editor, the action set editor, provides a visual programming environment to define story logic for each scene. This editor works with rules consisting of actions and conditions. Because the story logic can be constructed this way without the use of a programming language like many other story authoring tools, it enables people without programming experience to use the framework as well.

Another example is Scribe by Medler et al. [16]. Scribe is another authoring tool that aims to make story authoring easier by simplifying the process so no prior experience is needed. Similar to StoryTec's editors, Scribe has separate author modes displayed in separate windows. One of the author modes is element placement, where the author places elements like objects and characters on a 2D map that represents a 3D environment. Another author mode is story creation, where the author composes the story using a graph consisting of plot points.

Both tools managed to facilitate story authoring by combining all components needed to create a story into one framework. These components were simplified and made easy to work with to allow someone without any experience in digital story authoring or programming to create a story. Even though things were made less complicated, both frameworks encompass a lot of functionality. These tools, however, may not be the best option to make digital story authoring more accessible. Two elements could be improved. The first one is the interface. In both tools, functionalities are separated into different editors and there seems to be no direct integration between them. This can cause a disconnect, making it difficult for the author to connect the story logic to elements in the scene, for example. The second element up for possible improvement is visualisation. Both tools allow for a 3D environment to be used. Because they were developed for a desktop environment, these environments can only be displayed in 2D. Scribe even uses a representation the authors described as 2.5D, a 2D top-down view of the environment with height annotations. AR could provide improvements for these elements. It allows the author to view their story environment and placement of characters and objects in 3D. The scene can be explored by simply moving the head. The extra dimension in AR allows the interface to occupy more space. This spatiality can be used to integrate different components of the digital story authoring process into one view.

3 AR Interaction Design

Augmented reality combines the real world with virtual elements by providing a digital overlay on top of a person's view. As with any electronic device, an interface is needed to interact with these virtual elements. Because AR is not a completely separate element people interact with as it integrates into a user's surrounding environment, it is quite a unique piece of technology and therefore requires a very different approach for designing a convenient interface. Based on the related work discussed in the previous chapter, two prototypes were created as initial interaction design experiments, one for each interface type, to explore interaction possibilities. Both are described in Section 3.1. Using the principles and findings of these prototypes, interactions were designed for each interface type that can be combined to create an interface that involves both tangible and touch-less interactions. These are presented in Section 3.2.

3.1. Prototypes

To explore interaction options within both categories of interfaces, tangible and touch-less, two prototypes were created. Each prototype contained some basic operations like selection, movement and property changes. The tangible category was chosen as the starting point for the prototypes. The first prototype includes tangible interactions only. It is based on the simplest and most accessible form of tangible interfaces, marker tracking. The second prototype builds upon the first one by replacing tangible interactions by equivalent touch-less actions or changing them to different ones. Both prototypes were designed for a head-mounted device and a tabletop environment.

3.1.1. Tangible Prototype

Kato et al. [2] introduced an AR interface using markers, which are patterns on objects, most commonly looking similar to a QR code. The patterns are used to identify a marker and its position and rotation by a camera, so the AR device can display virtual elements accordingly. In the application developed by Kato et al., each marker displays one virtual element on top of it. When two markers with similar elements were placed close to each other, the virtual elements would interact. Later, this interface idea was expanded by Poupyrev et al. [4] in an attempt to create a generic AR interface using markers. In this interface, variable markers were introduced. These markers do not represent any virtual element at first but can be assigned one at run time from a catalog. Some action markers were added to perform operations on the variable markers like copying, deleting and showing info.

This interface was used as inspiration to build the first prototype for this project, especially the idea of variable markers combined with action markers used to modify the content of variable markers. This concept allows the user to decide what virtual elements are represented on markers, which makes the interface more adaptable than having each marker represent one virtual object.

Similar to the interface by Poupyrev et al., the tangible prototype contains variable markers and action markers. The variable markers do not contain any virtual content by default. However, because tracking was not robust, a small red square was added to each variable marker to denote when it was

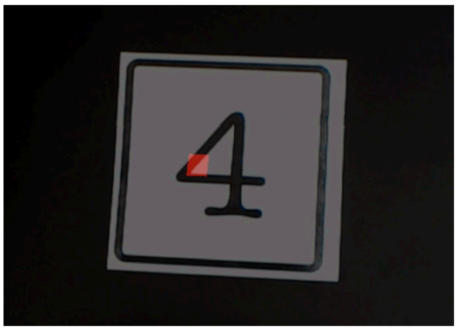
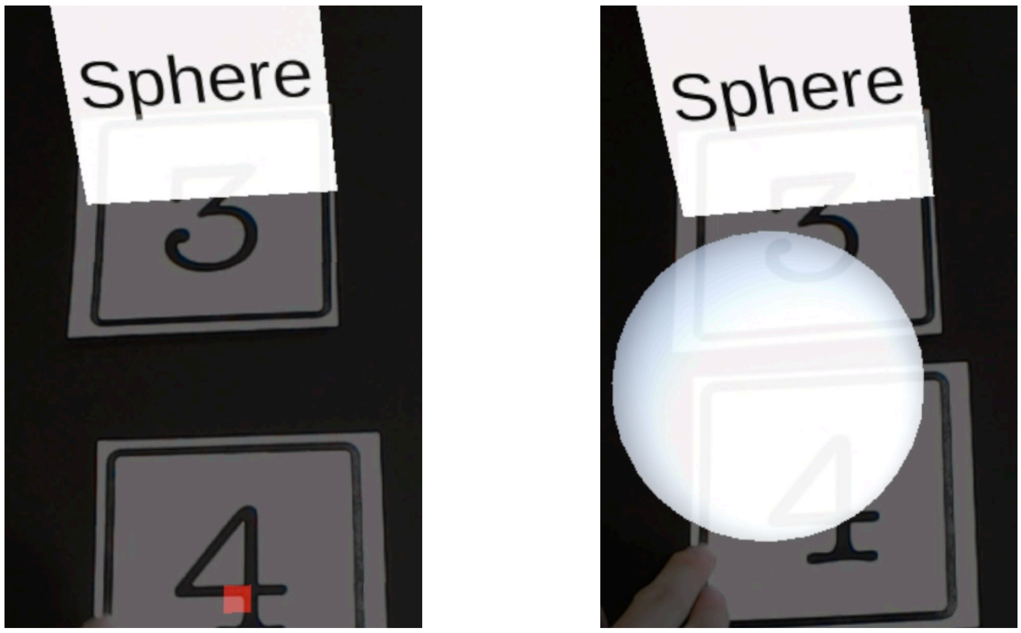


Figure 3.1: A variable marker in its default empty state.

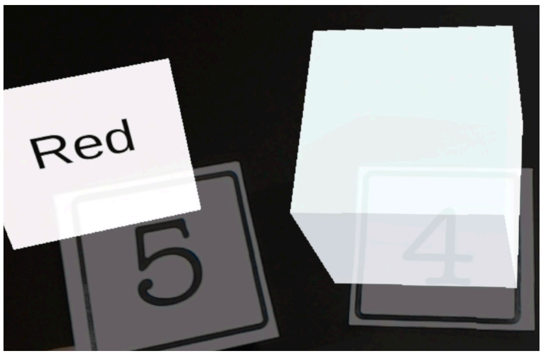
tracked by the camera while empty, as can be seen in Figure 3.1. There are 5 different action markers in this prototype. Two action markers add content to a variable marker, more specifically a cube or a sphere. When the cube or sphere action marker is placed close to an empty variable marker, the respective object spawns on top of and is attached to the variable marker. This action can be seen in Figure 3.2. Two other markers change a property of the content on a marker, colouring the object red or blue. This action can be seen in Figure 3.3. The fifth and final action marker represents a reset operation. Placing this marker close to a variable marker that represents any content, removes that content from the marker, which reverts it to its empty state as shown in Figure 3.4.



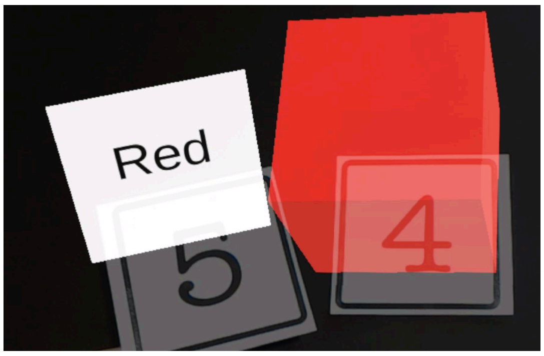
(a) An empty variable marker before moving close to the sphere action marker.
(b) The variable marker after moving close to the sphere action marker. It now contains a sphere object.

Figure 3.2: The action of adding an object to an empty marker.

With this setup of content markers and operation markers, multiple instances of the same element can be used and many different combinations of content and properties can be made. However, looking at the purpose of the application this prototype is built for, there is a scalability issue. For content creation applications like story authoring, it is preferred to have a considerable number of different elements available. This would result in a large number of markers, each representing a character or object to copy from to a variable marker. In other words, for this concept to work, there needs to be a way to group the available content. There needs to be some kind of catalog or menu that can be used to choose content from. It is possible, of course, to have a virtual menu that contains all the content options. This menu can be tied to a marker or can be stationary in the virtual environment. To use

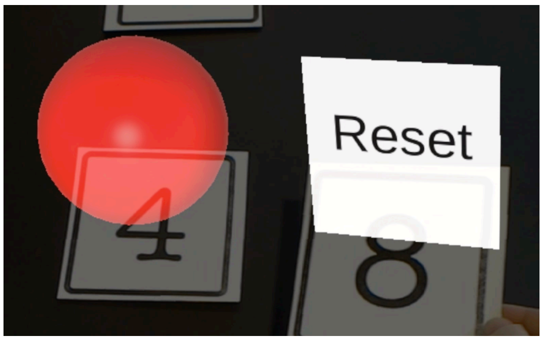


(a) A variable marker containing a cube object before moving close to the red action marker.

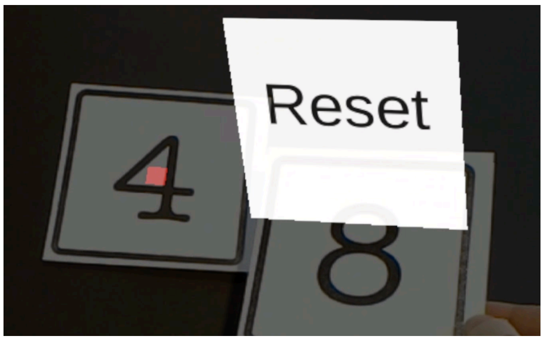


(b) The variable marker after moving close to the red action marker. The colour of the cube was changed to red.

Figure 3.3: The action of changing a property of content on a marker.



(a) A variable marker containing a sphere object before moving close to the reset marker.



(b) The variable marker after moving close to the reset marker. The sphere was removed from the variable marker.

Figure 3.4: The action of removing content from a marker.

this menu, a selection mechanic is needed. Considering markers only, it is possible to have a marker represent a virtual pen. There is, however, a second type of AR interfaces, as discussed in the related work chapter, that could be a better fit for this mechanic.

3.1.2. Touch-less Prototype

Hand interaction is considered to be one of the most natural ways to interact with AR content. It is how we interact with the world naturally which is why it makes sense to use it for virtual elements as well. In tangible interfaces like the first prototype, this is already the case in an indirect way. The user uses their hands to manipulate the markers which in turn manipulate the virtual content. With a touch-less interface using hand interaction, the virtual elements can be manipulated by the user's hands directly. The second prototype for this project is a continuation of the first prototype but was built using only hand interaction. Markers were made virtual. They were still present in the application and were used to anchor content to but the user could only interact with them using their virtual hands. Initially, this was because of software and hardware constraints but it resulted in some interesting discoveries.

This prototype contains two virtual markers and a vertical menu displayed in front of the user in the 3D space. Placed below the menu is a square that represents the marker programming space. This setup can be seen in Figure 3.5. When a marker is placed on the square, changes can be made to it using the menu. The menu contains two buttons, one labeled cube and the other labeled sphere. The user's hands are tracked and displayed as virtual hands. These virtual hands can be used to pick up and move the virtual markers and press the buttons. When a button is pressed while a marker is present in the programming space, the corresponding element is spawned on the marker. This can be seen in Figure 3.6. When the marker is moved away from the programming space, the spawned element follows the movement of the marker.

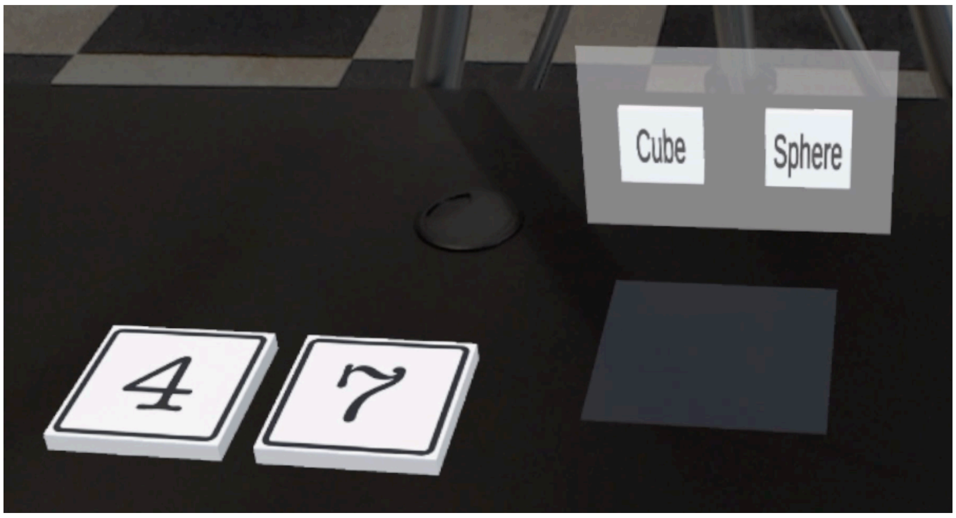
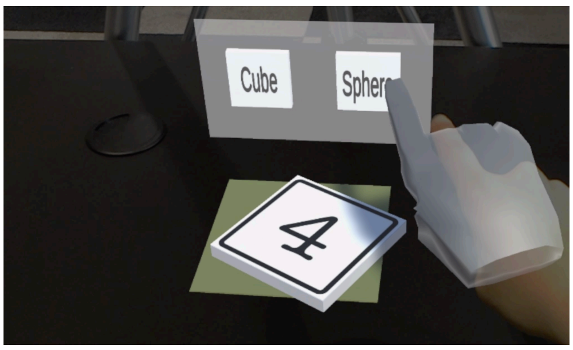
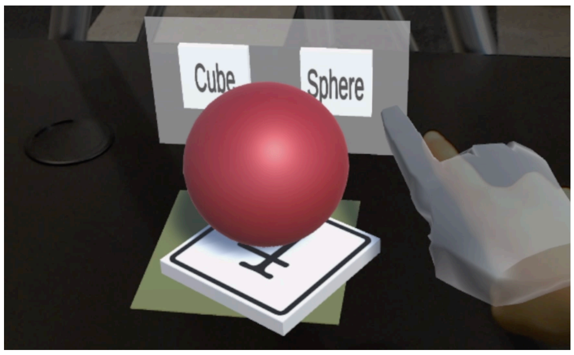


Figure 3.5: The setup of the touch-less prototype: a menu, a marker programming space and virtual markers.



(a) An empty virtual marker on the marker programming space before pressing the sphere button.



(b) The virtual marker after pressing the button. It now has a sphere attached to it.

Figure 3.6: The action of adding content to a virtual marker.

The menu contains only two buttons in this prototype but can easily be expanded into a full selection menu with many different options. The other actions from the first prototype, like changing colour, could be added as buttons as well. This keeps the mechanic of moving a marker close to another marker, in this case, the programming space, to add or change content while only requiring one menu instead of many different action markers.

Having all available options in one menu is a convenient way of programming markers. It does not require switching out markers for different operations. Furthermore, selection is simple. The user has to press a button on the vertical menu with their virtual hand which is equivalent to a pointing motion to the desired option. Moving virtual markers, however, was rather difficult in this prototype. Picking up a marker, using a grabbing motion, was not an easy task and placing it exactly on the targeted spot was often not successful. Although the hand tracking and physics implementation not being perfectly robust are a big factor, there is another reason why this will never be as natural as interacting with physical objects. Because touch-less interfaces are fully virtual, there is no tactile or haptic feedback. Without this feedback, the user can only rely on their vision to position their hand to correctly grab a virtual object. Even though the AR device used for this prototype supports a 3D view, it is difficult to pinpoint the exact location of something virtual. One reason for this is the lack of occlusion of virtual elements by physical objects. For example, when a virtual chair is placed behind a physical table, the chair will not be occluded by the table and instead be entirely visible, causing a distortion in depth perception. The same goes for a user's hands. When a hand is held in a position that is supposed to be closer to the user's head than a virtual object, the object is still displayed over the hand.

Focusing once again on the topic of this project, story authoring, it is clear that the exact placement of elements can be important. The author should be able to compose a scene with characters and objects, which includes placing these story elements in the scene on specific locations. Therefore, the physical markers from the first prototype suit this idea better. Conveniently programming markers with many different options for characters and objects is also an essential part of a story authoring application. The menu of the second prototype seems to be a good option for this. In other words, a combination of tangible and touch-less interactions is a viable option for a good interface for an AR story authoring application.

3.2. Combining Tangible and Touch-less

The design approach for this interface was not to start from nothing and completely invent a new AR interface by introducing concepts that have never been used before. Instead, it builds on existing techniques, capitalizing on what has already been developed and researched. Current research usually involves only one of the two types of AR interfaces, tangible or touch-less. As mentioned before, both types have their advantages and disadvantages. Combining interactions from both types in a useful way may result in improved interface concepts.

Limitations of the tangible and touch-less interface type have been discussed in Section 2.1 and some more strengths and weaknesses have been described in the prototype section above. By clearly defining what should be done by tangible interactions and what suits best with touch-less, each action can be assigned to the most fitting type. This can be done based on the strengths, weaknesses and findings mentioned before.

3.2.1. Tangible Interactions

Tangibles are physical, which gives them the advantage of tactile feedback and familiar physical properties. A virtual element tied to a marker follows the physical movement of that marker. This makes tangibles a good fit for spatial actions. A spatial action is any action that relates to the movement of a virtual element where placement in the 3D space is meaningful and has to be easily changeable. For a story authoring application, this translates to the placement of characters and objects in the scene. A character or object can be added to a marker and placed on the desired position in the scene by moving the marker to the position. Markers can be moved around freely to change the composition of characters and objects.

Because markers are very convenient to move, it is easy to move them out of and bring them back into the workspace. Reoccurring virtual elements can, therefore, be quickly removed from and re-added to the AR space. It does not require any additional operations like a selection from a menu. In story authoring, this can once again be translated to characters and objects. Especially characters are frequently reoccurring elements in a story. Once the characters are added to a marker, they can easily be added to the scene and taken away if they are not involved in the scene that is being authored at that moment.

Reoccurring elements are not always related to spatial actions like characters and objects. Markers can represent a general element or even operation that causes a change without being positioned. An example related to story authoring is the environment where a plot point takes place. If the story involves three locations, a living room, a kitchen and a garden, for example, markers can be programmed to represent each of these locations. When authoring a scene that is supposed to take place in the kitchen, the marker representing the kitchen environment can be brought into view, changing the environment of the scene. Where the marker is shown or placed is of no importance. As soon as the marker is registered, the environment of the scene changes. Furthermore, the marker does not need to stay in view. The environment can remain the same until another environment marker is moved into view.

Using markers for spatial actions and reoccurring elements means that, once each element is added to a marker, these operations only require the user to move markers around. One important factor of a tangible interface is that a surface is needed to place the markers on. The story authoring application for this project is designed for a tabletop environment which implies that there will always be a surface in front of the user where the markers can be placed within arm's reach.

3.2.2. Touch-less Interactions

As mentioned above, there needs to be a way to add virtual content to markers. This allows the author to dynamically assign the objects they want to use, chosen from a collection that is too big to have one marker per object, to markers of their choosing. Similar to the second prototype, a menu can be used to display all possible options for categories and marker content. A selection mechanic is needed to enable the user to choose the desired option. A very natural action to indicate a choice is pointing. Another familiar selection mechanic that relates more to technology is pressing buttons. This can be done with touch-less interactions using virtual buttons and hand tracking to enable the user's physical hands to interact with the virtual buttons. By displaying the options as buttons on a virtual menu, the user simply needs to point to what they want and perform a motion similar to pressing a button. Story authoring involves the selection of elements like characters, objects, actions, environments, properties and more. Using a menu to display all options with the described touch-less selection mechanic allows the author to quickly select story components from a wide range of choices.

With virtual buttons, there is, of course, no tactile feedback. To compensate, visual cues can be added to the virtual buttons, like movement to simulate the pressing of a physical button and change of colour to show selection. Because the AR application for this research is a tabletop application, the surface of the table can be used to provide a simulated sense of tactile feedback. Placing the menus horizontally on top of the table causes the user to touch the table when pressing a button, which is similar to a touch screen.

While tangibles are suitable for spatial actions and variable reoccurring elements, touch-less interactions can be used for interface elements that are static and do not have spatial importance. This includes general operations that can be translated to a button, a gesture or an interactive visualisation. An example in the case of story authoring is the storyline. The storyline should be available to the author for viewing and editing but where it is displayed is not of importance other than for user convenience. It should always be present or accessible to the author, making it a static element. Therefore, it would not make sense to make the author attach the storyline or plot points to a variable marker. Operations involving the storyline like adding a new plot point, browsing through plot points or going back to a previous plot point to edit it, can be implemented using similar concepts to the selection mechanic described above. Adding a new plot point can be represented by a virtual button that the user can press with their physical hands. A pointing and touching gesture to the desired plot point in the storyline allows the author to select it for displaying and possibly editing. A suitable and familiar interaction for browsing through plot points, especially when the storyline becomes rather lengthy, is a swiping gesture, similar to scrolling through a page using a touch screen. As before, tactile feedback can be simulated when displaying the storyline and the operation buttons horizontally on top of the table surface.

Many different operations can be implemented this way. Making virtual buttons represent operations keeps interactions consistent because it is similar to the selection mechanic. Buttons can be placed anywhere in the 3D space, or 2D space when placed on the tabletop the surface, possibly grouped by functionality for convenience.

To summarize, for all actions where the location in the 3D space matters and all actions that are variable and can reoccur, tangible interactions can be used. All actions involving selection and general operations where the location is not important can be done using touch-less interactions. An overview of each action category assigned to the suitable interface type can be seen in Table 3.1.

Tangible Interactions	**Touh-less Interactions**
Spatial actions Example: placing elements in a scene	Selection actions Example: selecting an item from a menu
Variable Reoccurring elements Example: changing a property of an element	General operations Example: scrolling through content

Table 3.1: Overview of interactions per interface type

4 Story ARtist Application

The Story ARtist application was developed to evaluate AR interaction and visualisation for story authoring. Because the aim of this project is to make story authoring accessible, the story representation is kept simple. To author a story, the author can create plot points, select actions and program characters, objects and environments on markers. How exactly this is done in the application is described below. The application's functionality will be discussed in Section 4.1 and the interface outline in Section 4.2. An example story can be found in Section 4.3.

4.1. Plot Point Structure

The overall structure of the story in this application is based on plot points. A storyline consists of a sequence of plot points, each one representing a single action. When a new plot point is created, an action is chosen and the plot point needs to be filled with information related to the action.

4.1.1. Actions and Arguments

An action is the kernel of each plot point. It is the main verb that represents what happens in the plot point. Authoring a story verb by verb could be tedious and would require the author to define many separate plot points. To avoid this and keep the focus on simplicity, the default actions that were chosen to add to the application are descriptive verbs that encompass multiple 'smaller' verbs that would be required to complete the action. An example is the verb *give*. In the application, this represents not just the action of a character handing over an object to another character. Instead, it includes the first character collecting the object, moving to the second character and handing it over, and the second character receiving it.

Once an action is chosen, it is assigned to the plot point and the interface displays what is needed in the scene to author the chosen action, called the arguments. For example, if the author chose the action *greet*, there need to be two characters in the scene where one character will greet the other.

4.1.2. Scene

In order to fill arguments, elements have to be present in the scene. In this scene, 3D models of characters, objects and environments can be added and arranged by programming them onto markers and moving the markers on the table. This mechanic allows the author to visualise how the authored plot point would look. It is a static representation of the action that can be seen as a snapshot of the story. When the author goes to the next plot point, which is when the plot point is complete and the author is happy with the scene arrangement, locations of the 3D models in the scene are saved in the application together with the action and arguments.

4.1.3. Plot Line

Because of the simplicity of the Story ARtist application and to keep the focus where intended, only linear plot lines can be authored and no consistency constraints are included. Plot points are displayed

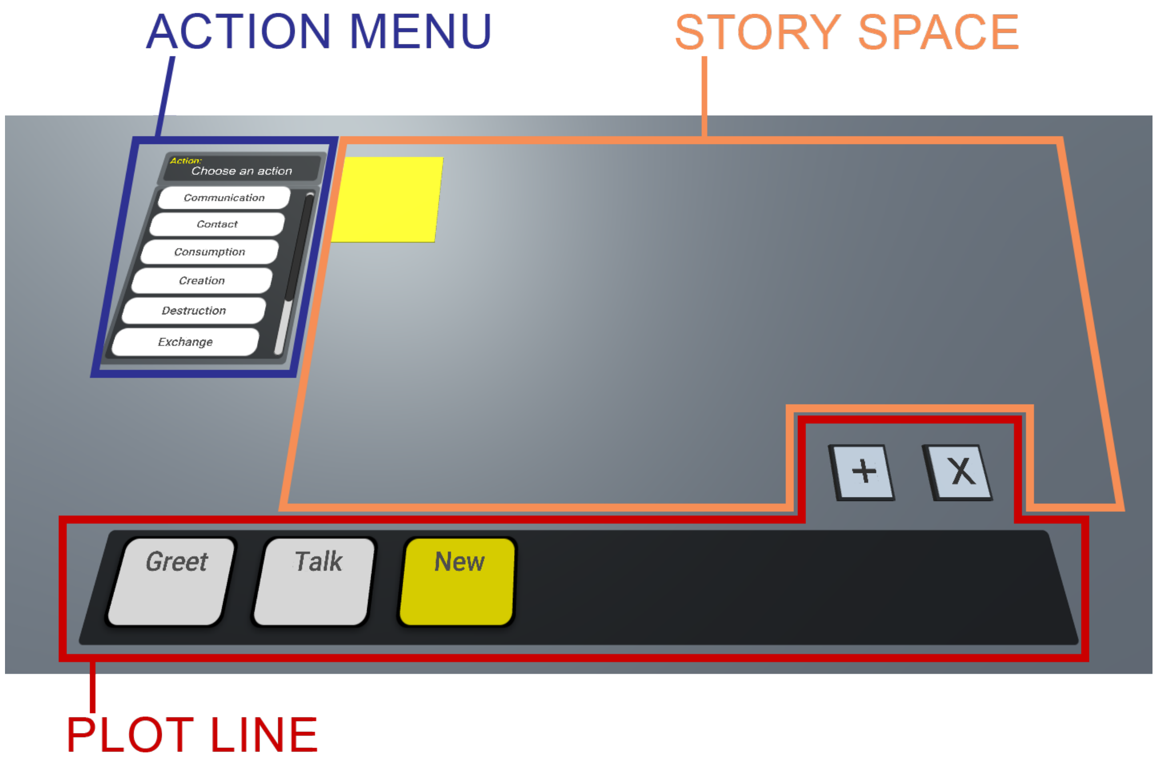


Figure 4.1: Overview of the Story ARtist interface.

as a linear sequence called the plot line. Selecting an existing plot point opens it, which allows the author to look at the authored scene with the chosen action, elements and scene composition. If the author wishes to make changes, they can edit a previously authored plot point after opening it. A new action can be chosen, the environment can be changed and characters and objects can be moved, added or removed.

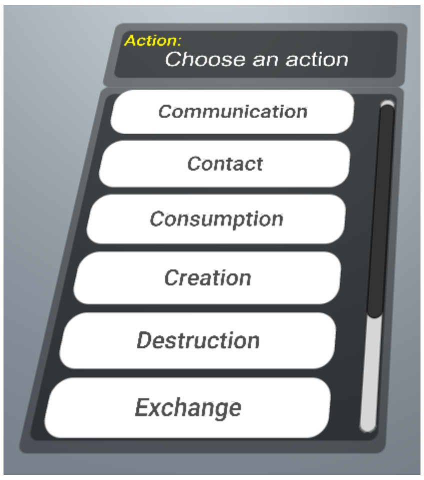
4.2. Interface

Three areas can be distinguished when looking at the application field as a whole. All three contain touch-less interactions with menus and/or buttons, which will be described in detail below. One area contains tangible interactions using markers which will be discussed separately in Section 4.2.2. All menus were designed using basic interface design principles like simplicity, visibility and error prevention, and 3D interactions using specific 3D interaction techniques [17].

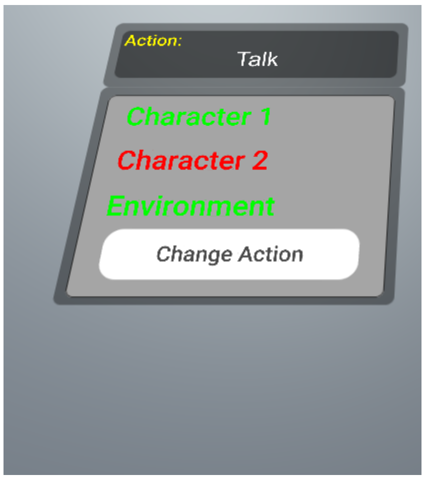
4.2.1. Interface Areas

The interface can be divided into three areas based on functionality: the action menu, the story space and the plot line. This can be seen in Figure 4.1.

The action menu can be found in the top left corner. It can be used to choose an action for each plot point. Actions are divided into categories for scalability. When an action button is pressed, the menu changes to display the action's arguments. Figure 4.2 displays the menu showing action buttons and arguments. The interface keeps track of what arguments are present in the scene and for which ones an element needs to be added. Arguments that are missing in the scene will be coloured red in the menu while arguments that have been filled in will be coloured green, which is how the author knows what still needs to be added to the scene to correctly complete the plot point. A *back* button is included in case the author changes their mind about the chosen action and would like to select a different one. When returning to the argument view with a different action, the arguments are refilled automatically with the elements that are present in the scene. For example, if the author chose the action *drink*, filled



(a) The action menu displaying the action categories as buttons. Physical hands can be used to interact with the buttons.



(b) The action menu displaying the arguments for the action *talk*. The colours show that a character and an environment are already present in the scene, and a second character needs to be added.

Figure 4.2: The action menu.

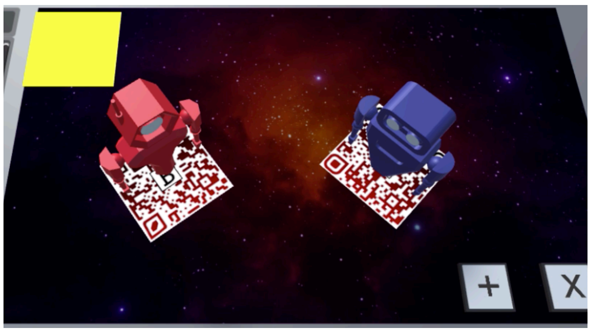


Figure 4.3: Example of an authored scene with two characters and an environment.

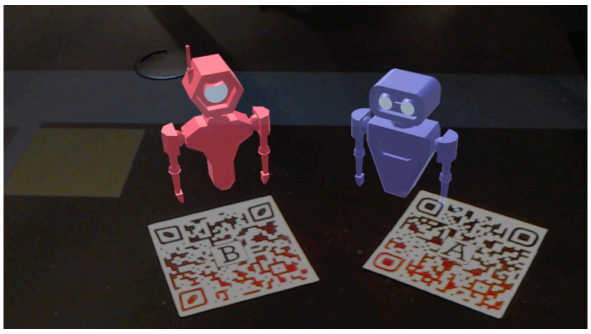


Figure 4.4: Example of an authored scene viewed through the AR device.

with a character and object and then decides to change the action to *eat*, the application will automatically recognise the character and object in the scene and assign them to the new arguments.

The story space is the 3D representation of plot points. This is where story elements like characters and objects can be placed in the scene to fill arguments and visualise the plot point. The story space contains a marker programming space, visualised as a yellow square, where content can be programmed on markers to place them in the scene. An example of an authored scene is displayed in Figure 4.3. Because it is difficult to obtain clear images through the AR device, most images from the Story ARtist application are taken from the PC view showing only the graphics. To illustrate what users see through the AR device, an authored scene captured through the device is shown in Figure 4.4.

While the action menu and story space are used to define the content of each plot point, the plot line area is where plot points can be controlled in their entirety. The plot line is displayed at the bottom of the application field as an ordered linear collection of all authored plot points. Plot points are visualised as buttons labeled with the action that was assigned to them as shown in Figure 4.5.

Above the list of plot points are some buttons corresponding to different plot line related operations. The *add* button, labeled with a plus sign, adds a new plot point at the end of the list and opens it to enable the author to add content. The *delete* button, labeled with an X, removes the plot point that is currently selected from the plot line.

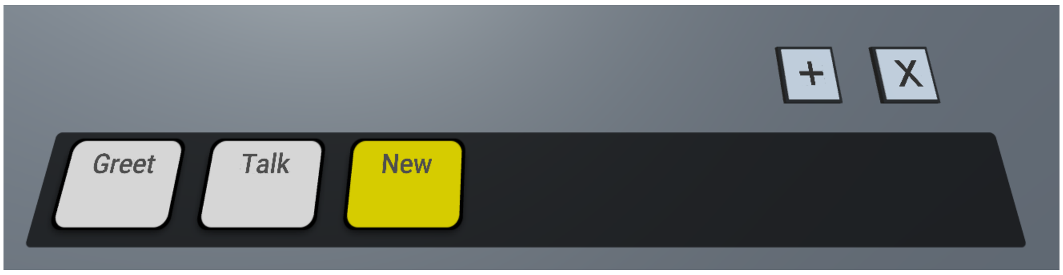
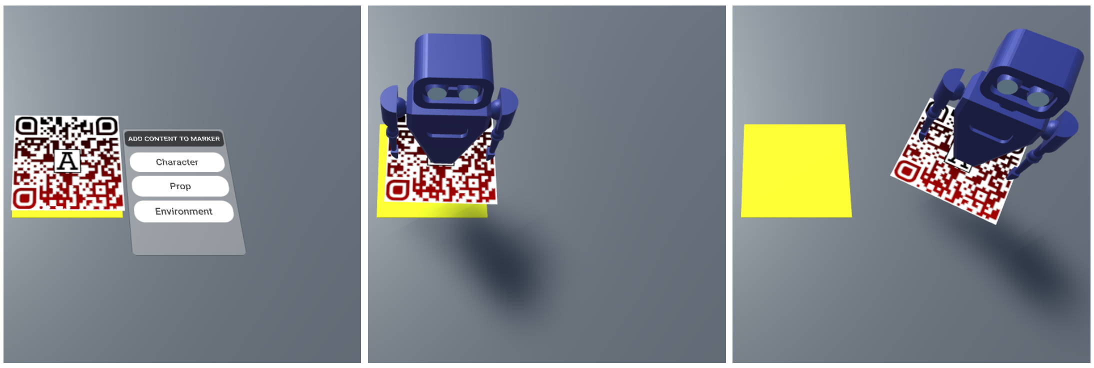


Figure 4.5: The plot line containing two authored plot points and one newly opened plot point.

4.2.2. Marker Programming

As discussed in chapter 3, markers are most suitable for spatial actions and variable reoccurring elements and should be variable to allow the choice between many options. Adding characters or objects to a scene is a spatial action because the placement of the element in the scene matters. An environment is a variable reoccurring element. Therefore, both types of interactions are done using markers. All markers that can be used in the application are variable, i.e. do not represent any content when starting the application, and a programming mechanic is included to add chosen content to markers.

As mentioned above, a yellow square can be found in the story space which represents the marker programming space. When a marker is placed on top of this square, a menu pops up that contains buttons with the available categories of content that can be programmed on the marker and can be used to choose the element that the marker will represent. The categories that are currently included in the application are character, object and environment. An example sequence of adding a character to a marker can be seen in Figure 4.6. Even after being programmed with content, markers remain variable. Their content can be changed by placing the marker back in the marker programming space and choosing a different element from the menu.



(a) When a marker is placed on the yellow square, the menu pops up to choose content to add to the marker. (b) The menu was used to select a robot character which is now programmed on the marker. (c) The marker can now be moved around in the scene to place its content on the desired location.

Figure 4.6: The sequence of adding content to a marker with the marker programming space.

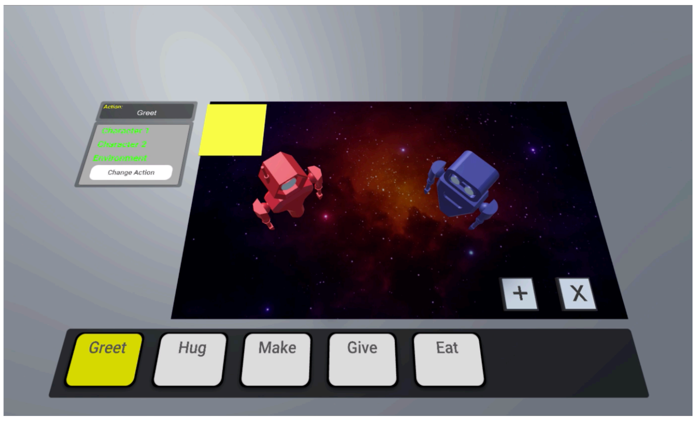
Characters and objects that are not yet in the scene can be added by programming them on a marker, if this was not done already, and moving the marker into the scene. Elements can be removed from the scene by taking their marker out of view. For a marker representing the environment of the scene, its location has no spatial importance because the environment takes up the entire story space. It suffices to show an environment marker anywhere in the application field until it registers and the environment changes. The scene's environment will not change, even when a new plot point is created, until a marker with a different environment is shown. This is because multiple consecutive plot points often take place in the same environment.

When an existing plot point is opened, the corresponding scene is loaded. As a result, elements can be present in the scene without their marker or can be in a different location than their marker. If the marker is not present in the scene, reintroducing it will place the virtual element back onto the marker.

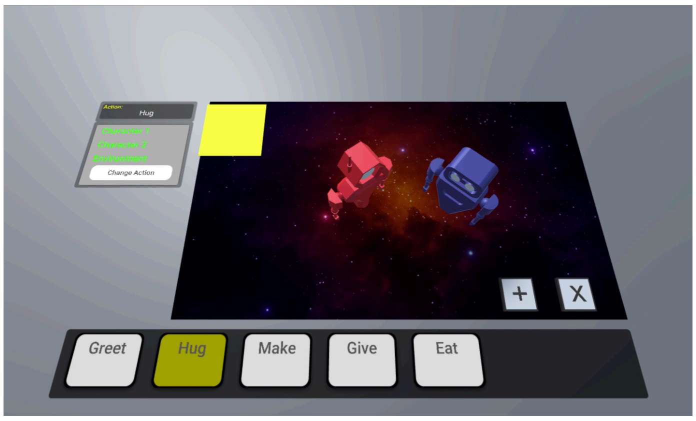
This way, the element can be controlled again so it can be given a different location in the scene or removed from the scene. If the marker is present in the scene but its content has moved because it was placed on a different location in the plot point that was opened, moving the marker will also reattach it to the marker to make it controllable again.

4.3. Example Story

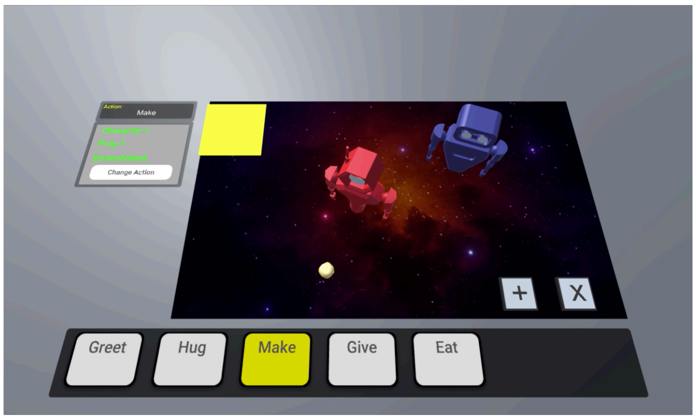
Once the author is done creating and all plot points have been authored, the story can be ”played out” by going through each plot point by opening it, to see each scene after the other. An example of a resulting story authored with the Story ARtist application is shown below.



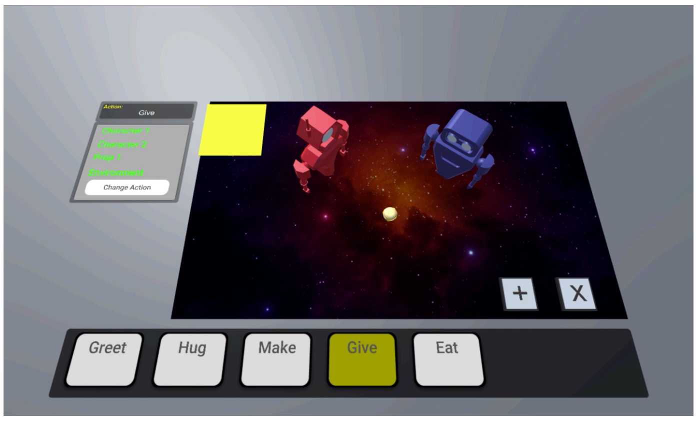
In the first plot point, the action *greet* was chosen. It required two characters and an environment, which, as can be seen in the image, are present in the scene. The red robot and blue robot greet each other.



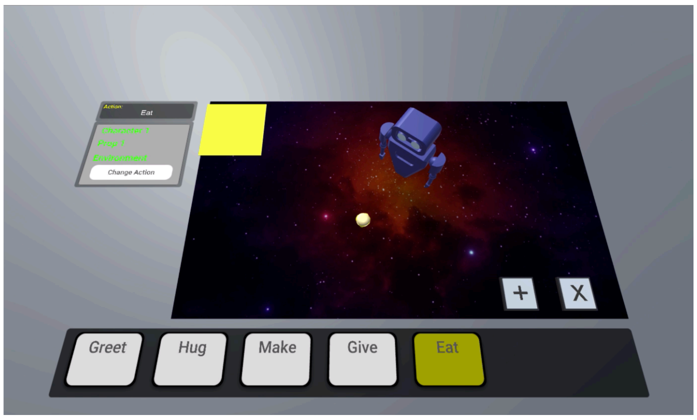
The next plot point contains the action *hug* which has the same arguments as the action in the previous plot point so nothing new had to be added. Because it makes more sense for the story, more specifically for the chosen action, the robots were placed closer towards each other.



For the action *make*, a prop, a burger to be more specific, was added to the scene. Even though the action requires only one character to be present in the scene, a second one can be present because not all elements in the scene have to relate to the action.



The next plot point contains the action *give*. All required elements were already present. Similar to all previous plot points, the elements were positioned to reflect the chosen action, which, in this case, visualises the red robot giving the burger to the blue robot.



The final plot point was authored with the action *eat*, which requires one character, one prop and an environment. Although not necessary, the author decided to take the red robot out of the scene, leaving the blue robot to eat the burger.

5 Implementation

After delineating the front-end side of the application in the previous chapter, this chapter focuses on the back-end side, describing everything needed to run the application and how the application works internally. Section 5.1 describes the application setup used when developing and evaluating the application. This includes all hardware devices and software libraries. The framework used to represent the story in code and store it in data files is explained in Section 5.2. In Section 5.3, the software architecture of the entire application is presented.

5.1. Setup

Because AR is a relatively new kind of technology, advanced AR technology is not very accessible yet. Furthermore, the devices that are available are not easily adaptable for every application. Therefore, it requires some creativity to create a setup that includes all the features needed for an AR project. In this section, all hardware and software used to build the setup for the Story ARtist application are described.

5.1.1. Hardware

Initially, this project started with an AR device and a PC. It quickly became clear more hardware was needed to not let the research be limited by the technology. Below is the list of hardware devices used for the Story ARtist setup. A visualisation of the setup can be seen in Figure 5.1.

AR Device

An important hardware requirement for this project was to use a head-mounted AR device to enable better interaction. Other accessible device options for decent AR at this moment are smartphones or tablets which are handheld devices. Using a head-mounted device enables the user to use both hands to interact with the AR interface. The available AR device for this research was the Microsoft Hololens, first generation [1].

External Camera

The Hololens has a regular camera built in that can be used for marker tracking. This, however, was not optimal. Markers had to be held close to the camera for a considerable amount of time for them to be recognized. This was a problem when markers were placed on the table and therefore viewed from an angle. To solve this problem, an external camera was added, a Logitech C920 Full HD Webcam. The camera is placed above the table, pointing down. The tracking improved because the camera was stationary and the angle between the camera and the markers placed on the table was reduced significantly.

[1]Microsoft, Hololens Hardware, https://docs.microsoft.com/en-us/hololens/hololens1-hardware

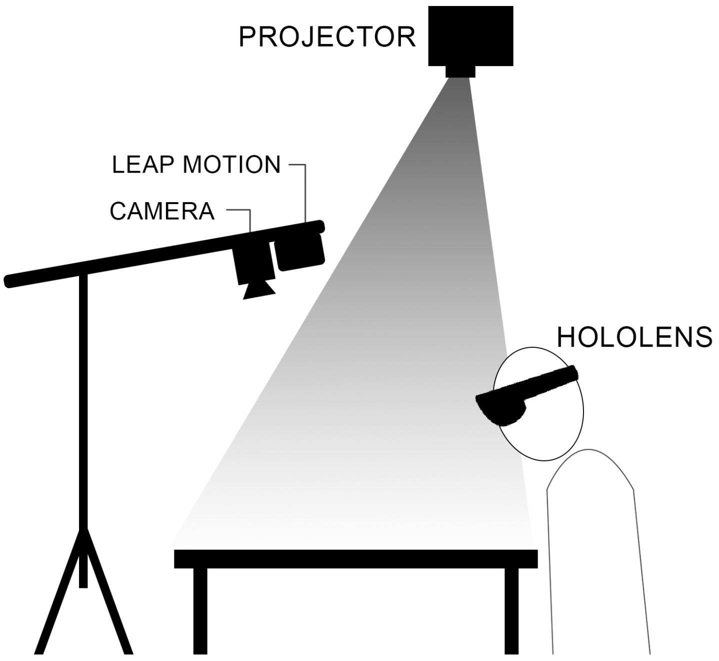


Figure 5.1: Visualisation of the hardware setup for the Story ARtist application.

Hand Tracking Device

The interface designed above includes touch-less interactions, hand tracking to be more specific. By default, the Hololens supports very little hand tracking and gestures. To avoid being limited by the lack of touch-less interaction supported by the Hololens, a Leap Motion Controller [2] was added to the setup. This is a small device specifically designed for hand tracking. The Leap Motion was placed on top of the external camera above the table, pointing down like the camera. It is worth noting that this device could be discarded in future setups since hand tracking is often being integrated into current head-mounted AR devices. This includes the second generation of the Microsoft Hololens.

Projection

A big problem faced with head-mounted AR devices at the moment is the field of view (FOV). It is not yet feasible to provide head-mounted AR devices with large FOV. The Hololens' FOV is 30 by 17 degrees which is a considerably small portion of our natural FOV. To counter this, projection was added. A projector was placed on the ceiling, pointing down to the table so a projection overlay could be added to the tabletop environment.

PC

All devices were connected to a PC that runs the application. An Alienware Aurora 2012 PC was used in the final setup with a GeForce GTX 590 graphics card. The Hololens uses a wireless connection to communicate with the application running on the PC. To improve this connection, an Asus AC3100 WiFi adapter was added.

5.1.2. Software

To implement the StoryARtist application, a game engine was chosen. Game engines support many libraries and tools which are needed when creating an AR application that involves many different elements and hardware like StoryARtist. To integrate all hardware devices, use both tangible and touch-less interactions and provide a storage system for data and authored stories, several libraries were added to the project.

[2]Ultraleap, Leap Motion Controller, https://www.ultraleap.com/product/leap-motion-controller/

Game Engine

The application was built in Unity 2018.4 [18]. This is a Unity version with long-term support that many libraries support as well, including the libraries needed for the hardware used in this project. The Unity editor, more specifically Unity's Play Mode, was also used to run the application. This was the only feasible way to integrate all hardware into one application.

Libraries

Ultraleap provides different libraries to integrate their hand tracking software into a Unity project. For this project, the core SDK for Unity [19] was used together with the interaction engine, which enables interaction with custom UI elements.

For tangible interactions, a library was needed to track markers. This functionality was added using the Vuforia Engine [20], an AR library that tracks markers and objects using image processing. The Vuforia engine was configured to run on the frames provided by the external camera.

Data management for both story assets and authored stories as a result of the application was done using JSON. The Unity asset *JSON .NET for Unity* [21] was added to the project for easy integration of JSON parsing into scripts.

5.2. Story Framework

Apart from interaction and visualisation, another challenge arises when developing a story authoring application. The story needs to be represented internally in code. As explained before, a plot point structure was designed to use as the foundation of the story framework, which is described in more detail below.

The story framework is split up into two sides. One side is the internal representation of the story that is used to store plot points in the application so the authored data can be converted to a file in a common format that can be used outside of the application. The other side is the representation that is application specific and is mainly used to visualise the story.

5.2.1. Representation of the Story

Story ARtist is a concept application developed specifically for creating a simple linear narrative that can be used as a baseline for a story. To further develop the story and its elements, it is useful if the plot line written with the application can be exported for use with other tools. To enable this, a common framework was used as inspiration for the story representation in the application and a widely used format was chosen to store the plot line in a file.

Many applications that involve some kind of narrative use actions, or a grouping of actions, as base units for the story. Actions, or verbs, are a good identifier for story events as they are often unique as opposed to characters, for example, which usually reoccur in multiple consecutive events. Therefore, verbs were chosen as a base for each plot point, with arguments that identify what the verbs require.

What arguments each action requires is defined by its predicates. In modern English syntax and grammar theories, predicates are defined as the main verb in a sentence together with any auxiliary verbs associated with the main verbs. In this story representation, predicates are the subdivisions of the main verb/action, each representing an event that is part of the main action. When combined, they form the entire main action. For example, the predicates of the verb *give*, where one character gives an object to another character, would be *grabbing the object*, *moving to the other character* and *transferring the object*. Similar to the definition in syntax and grammar, the main verb can be one of the predicates. An example is the verb *eat* of which the predicates are *grabbing the object* and *eating the object*.

This representation of actions, predicates and arguments was inspired by VerbNet [22], an English verb lexicon. VerbNet is often used for natural language processing but can also be used for story-related contexts like computational storytelling [23, 24]. Two recently presented updates to VerbNet on generative lexicon event structures [25] and subevent semantics of transfer verbs [26] describe the use of predicates to divide verbs into more specific fragments, defining its subevents in detail. This representation was adapted to fit the story ARtist framework, as described above.

To store the authored plot line to enable easy export for external use, the JSON (JavaScript Object Notation) format is used to write the plot points to a text file. JSON is a lightweight data-interchange format that is easy to parse and generate in code while being readable and writable for programmers as well. Plot points are stored as a collection of the chosen action, the filled arguments, the location and other elements in the scene. Other elements placed in the scene during authoring are included because not all elements present in a scene may relate to the main action of the plot point. This does not imply, however, that the unused elements should disappear from the plot point. Using the JSON format together with the VerbNet-inspired structure for defining an action's arguments, enables easy conversion of the authored plot line to other tools, especially when the VerbNet representation is used.

5.2.2. Visualisation of the Story

While predicates define what arguments an action needs in the story representation, they are not used for the application's internal representation or storing of the authored plot line. However, apart from use in other tools, they can be very useful in an adaptive framework for the animated visualisation of the story. For every predicate, an animation can be programmed. For example, for the *move* predicate, the programmed animation is the first argument, a character, moving to the location of the second argument, which is either a character, object or other. As described in Section 4.1, the verbs used in the Story ARtist application are used as global actions where the framework automatically includes smaller subdivisions of the verb, i.e. predicates, to make the action work. Because of this, verbs often contain the same predicates. Every verb that requires a character to be close to another character or an object, will contain the *move* predicate to move the character close to the other character or object. This way, a list of different predicates can be programmed which can then be used to compose a wide variety of actions to include in the application and allows for easy adaption of the actions, including the addition of new options.

5.3. Software Architecture

The Story ARtist application was developed using the model-view-adapter (MVA) design pattern. This pattern separates the interface from the story data with a mediating controller in between, which characterises the overall structure of the implementation. For the story representation classes and mediating controller, a specific structure was used to separate how the story data is represented inside the application for storage and how it is represented for visualisation. An overview of the software architecture can be seen in Figure 5.2. An independent data system was used to manage the resources needed in the application.

5.3.1. General Implementation Structure

The MVA pattern used as the foundation for the application's implementation is a variation of the model-view-controller (MVC) pattern that is commonly used when implementing a user interface. The MVC design pattern has a triangular communication structure between its three components, the model, view and controller. The only difference between the MVA and MVC design patterns is that MVA is arranged linearly such that there is no direct communication between the model and view. In Story ARtist, the view corresponds to UI elements, the model corresponds to the scripts that contain the story representation and the adapter acting as a mediating controller is called the *application manager*. The UI elements communicate the user's input to the application manager. Based on this information, the application manager communicates with the code responsible for the story representation to pass on what needs to be changed in the story data or structure and possibly retrieve information about the story data. The application manager then coordinates changes in the UI elements based on user input or retrieved data.

5.3.2. Separating Story Framework and Visualisation

The story framework is used to represent the story using a plot point structure by storing the action, arguments, environment and elements in the scene for each plot point. However, to visualise the story as 3D scenes, different information needs to be stored. For example, the location of the 3D object of a character in the application's 3D space needs to be stored for visualisation while it is of no use

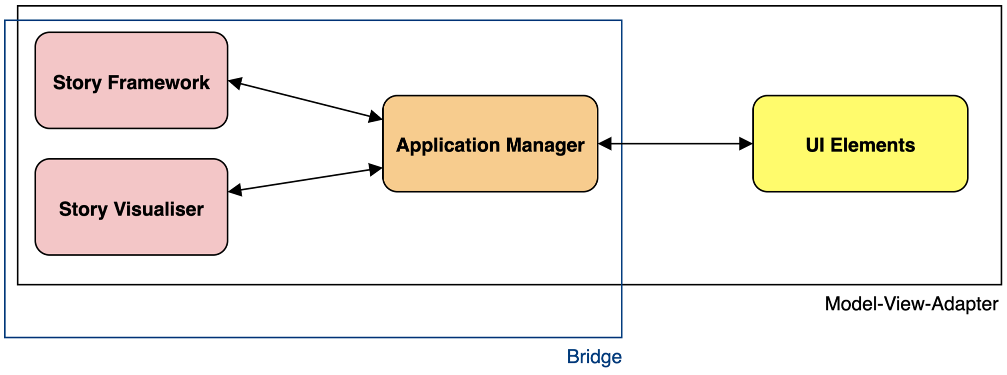


Figure 5.2: Overview of the software architecture.

to the story framework. Separating this information by creating a second story framework specifically for visualisation allows for independence and interchangeability. Therefore, the bridge design pattern was used to implement both types of story frameworks with the application manager. This pattern was designed to separate implementation and interface which corresponds to separating the story representation and visualisation in this case. The visualisation framework is called *story visualiser* and is used to store all story data needed for the application's visualisation.

5.3.3. Resource Management

When it comes to the management of data and resources that the application requires, two important factors need to be considered. The first factor is adaptability. To make the application adaptable, it should be easy to add new resources for every available category, i.e. characters, objects, environments and actions, and change or remove existing ones. The second factor is the scalability. As mentioned in the chapters before, the author should have a wide variety of options to choose from to use in their story. When a large number of resources needs to be loaded when starting the application, this could cause a long wait. To suit both factors, a resource management system is needed that can easily be adapted and can load resources at run time, only when they are requested.

Story ARtist's available resources are all stored textually in JSON data files. They can be found in Appendix A. This enables easy parsing in code and adaptability as it is easy to read and write JSON objects both in code and manually. For all available categories of elements, a data file is present that contains the names of all available elements divided into element-specific categories. In the actions data file, the predicates are mentioned as well. All other categories have corresponding 3D elements that need to be loaded on demand. Those data files contain the path to the location of their 3D resource. All data files are loaded upon starting the application. At this point, resources are stored as text only. When a 3D resource is requested, it is loaded from outside the application using the path stored in the data file.

6 Evaluation

Using the Story ARtist application and its setup, a user study was conducted to evaluate AR for story authoring with the proposed interaction techniques. A formative evaluation producing mostly qualitative data was chosen because it best suits the prototype nature of this application and gives a good insight into participant feedback and ideas. The user study setup and procedure are described in Section 6.1. How the data was processed is described in Section 6.2, and quantitative and qualitative results are presented in Section 6.3.

6.1. User Study

A total of 20 participants came by, each for an individual session of 45 minutes. All participants were students, recent graduates or postdocs. The session included an introduction to the application in the form of a narrated tutorial, a task-based interaction phase with the application, and an interview. Each participant interacted with the application for 10 to 20 minutes during the tutorial and interaction section. When interacting with the application, the participants were observed and the graphics view of the application on the PC monitor was recorded.

Upon entering, participants were given a short introduction about the project and a brief explanation of the setup they were about to interact with, followed by the procedure of the user study. Next, the tutorial was started. Because of the many devices involved in the setup, the technology of the Story ARtist application was not perfectly robust all the time which caused some issues. For example, hand tracking was often lost which required the participant to hold out their hand close to the sensor to resume tracking before continuing with operations. Because of this, the participants were asked not to focus on the issues caused by the technology upon starting the tutorial. Instead, they were asked to focus only on the application and its functionality, as technology-related problems are not part of this research.

During the tutorial, participants were guided by the observer through the operations required to author one plot point. This included selecting an action of their choosing and composing a scene based on the action's arguments by programming markers.

After the tutorial, participants were given the task to further explore the application by authoring some more plot points by themselves. A list of all operations the participant was required to perform was kept by the observer to make sure every participant tried all significant functionality related to story authoring. Once the participant was done adding plot points, they were asked to perform the remaining operations. Even though people were made aware that there was no time limit to author more plot points and were told to experiment with the interface, many simply authored plot points without further exploration. This caused them to completely miss out on other functionality, like going through previous plot points to review the scenes and possibly edit them. This was prevented by keeping the list of required operations and asking the participant to try out the functionality they did not explore yet.

When the participant was done interacting with the application, the interview followed. The interview was semi-structured with mostly open questions to explore participants' opinions without being

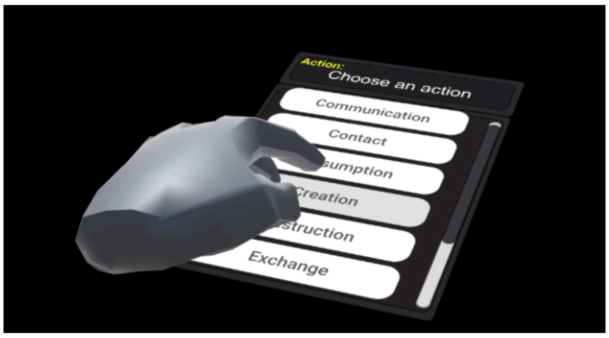


Figure 6.1: A virtual representation of a physical hand interacting with virtual buttons. Ideally, the user can use their natural hands to interact with buttons, without a virtual representation.

restricted by options. A list of 11 interview questions, which can be found in Appendix B.1, was composed, which focused on the main aspects to be evaluated. Each participant was also asked to describe what kind of previous experience they had with augmented and virtual reality.

Finally, the participants were given a questionnaire in the form of an online survey that could be completed via smartphone. The chosen questionnaire was the System Usability Scale (SUS) [27], to evaluate the application's usability. More specifically, the updated version by Bangor et al. [28] was used. It consists of 10 statements that participants had to give a score from 1 (strongly disagree) to 5 (strongly agree). The statements can be seen in Appendix B.2. Even though it is recommended to do questionnaires before interviews to avoid bias, the questionnaire was done last because of time constraints as it did not require involvement from the observer.

Because of the technological issues mentioned earlier, some changes had to be made to the Story ARtist application to make sure it was convenient enough for a user study. Ideally, users would control the application with their physical hands and press the buttons that were projected by touching the table. This was not possible as it was not feasible to calibrate the system for every user study participant. Instead, virtual hands were displayed that followed the user's hand movements closely but were not perfectly aligned with the physical hands which made controlling them more difficult. Figure 6.1 illustrates what the virtual hands looked like. To enable the user to better assess where to place the virtual hands to press buttons, all interface items that were originally projected onto the table, now had to be displayed through the Hololens. Only the marker programming space was projected so markers could be placed on top of it without a hologram blocking the marker because of the lack of occlusion. As a result, the FOV was once again limited to the small screen of the Hololens for all interface elements except the marker programming space.

6.2. Data Processing

Three different kinds of data were collected: observation data collected from the recorded videos, interview answers and questionnaire answers. Because the SUS was used, processing the questionnaire answers was done according to the SUS procedure. To process the qualitative data from the video recordings and the interviews, and draw the appropriate conclusions, the affinity diagramming process for evaluating interactive prototypes introduced by Lucero [29] was used.

Affinity diagramming is a technique commonly used in qualitative analysis to organize large amounts of unstructured data into hierarchical clusters to identify recurring patterns or themes. Beyer and Holtzblatt [30] introduced the use of affinity diagrams for contextual design where it was used in one of the initial stages of product design. Now, it is often used in human-computer interaction design for different purposes, including the evaluation of interactive prototypes. This makes it suitable for this evaluation because of the prototype nature of the project. An affinity diagram allows ideas to emerge from the data without any predefined categories or themes. As a result, opinions and ideas of participants can be explored without the limitation of predefined options. This is valuable because this research is a first look at how AR could be used for story authoring.

Following the process designed by Lucero, the observations from the recordings and interview answers were first converted into individual notes. 580 notes were created in total. As suggested by Lucero, each participant was assigned a specific colour to easily spot how many people mentioned a similar idea or issue. Next, notes were grouped forming clusters about a similar general topic and sub-clusters about more specific issues within a topic. These clusters can be used to identify common patterns, ideas and problems. An overview of the affinity diagram can be seen in Figure 6.2.

For qualitative evaluation techniques like the affinity diagram, it is recommended to have multiple people process and analyse the data to avoid bias and verify conclusions. The evaluation for this paper, however, was done only by one researcher.

6.3. Results

6.3.1. Quantitative

Following the SUS processing method, all scores from the questionnaire were processed resulting in a usability score per participant between 0 and 100. The distribution of the usability score over participants can be seen in Figure 6.3, which shows that most participants gave the application a usability score between 81 and 90. The average usability score is 78.62 with a standard deviation of 9.94.

Because people's experience with VR and AR was recorded, it is possible to compare the usability scores with this experience. To do so, each participant was given a score from 0 to 5 according to their VR/AR experience. A score of 0 corresponds to no experience and a score of 5 corresponds to owning a premium VR or AR device with levels in between distinguishing between, for example, mobile VR and more complex VR systems with controllers or hand tracking. The results can be seen in Figure 6.4. This graph does not show a clear trend of how the usability scores relate to previous experience with VR and AR. This means that overall, not only people who are familiar with VR or AR but also people who barely tried it found the application to be relatively usable.

6.3.2. Qualitative

As can be seen in Figure 6.2, out of the notes created for the affinity diagram, 7 clusters emerged of which 5 are worth discussing in detail. The remaining two clusters are about the UI and technical issues. Participants raised general issues regarding the UI, e.g. the buttons were too small, the scroll bar was confusing and visual feedback from buttons was not always clear enough. Regarding technical issues, 13 participants mentioned the difficulty of the technology, issues with hand tracking or issues with marker tracking during the interview. The other clusters are discussed below.

General Reactions

Overall, reactions were positive with many describing their experience as fun or interesting. People described the application as a good concept for visualisation, more specifically that it is helpful for expressing creativity or rapidly visualising a simple idea for a story. Some others mentioned the application concept to have potential for children and education. The majority, 17 out of 20 participants (17/20), thought the Story ARtist application was a good concept for story authoring in AR.

Interaction

The largest number of notes related to interaction. They were grouped in 3 sub-clusters: markers, hand interaction and the combination of both. From the notes in the combination cluster, it was clear that a majority (14/20) found the combination of tangible and touch-less interaction useful. People expressed that having markers for placement and hand tracking for selection is a good combination, that it belongs together and it does not create a disconnect. Out of these 14 people, 4 noted that if hand tracking would work perfectly, markers would no longer be necessary. Those who did not find the combination favorable expressed that they would prefer hand interaction only or the addition of speech.

Though 14 people liked the interaction combination, more participants (17/20) noticed the advantages of markers. Many remarks were made about the ease of use to position and rotate elements. Some participants related this to the physical aspect of markers, claiming that it is good to have something physical to manipulate and touch. Some participants compared the way markers were used here to playing with toys as a child, positioning them when imagining a story. A limitation of variable markers

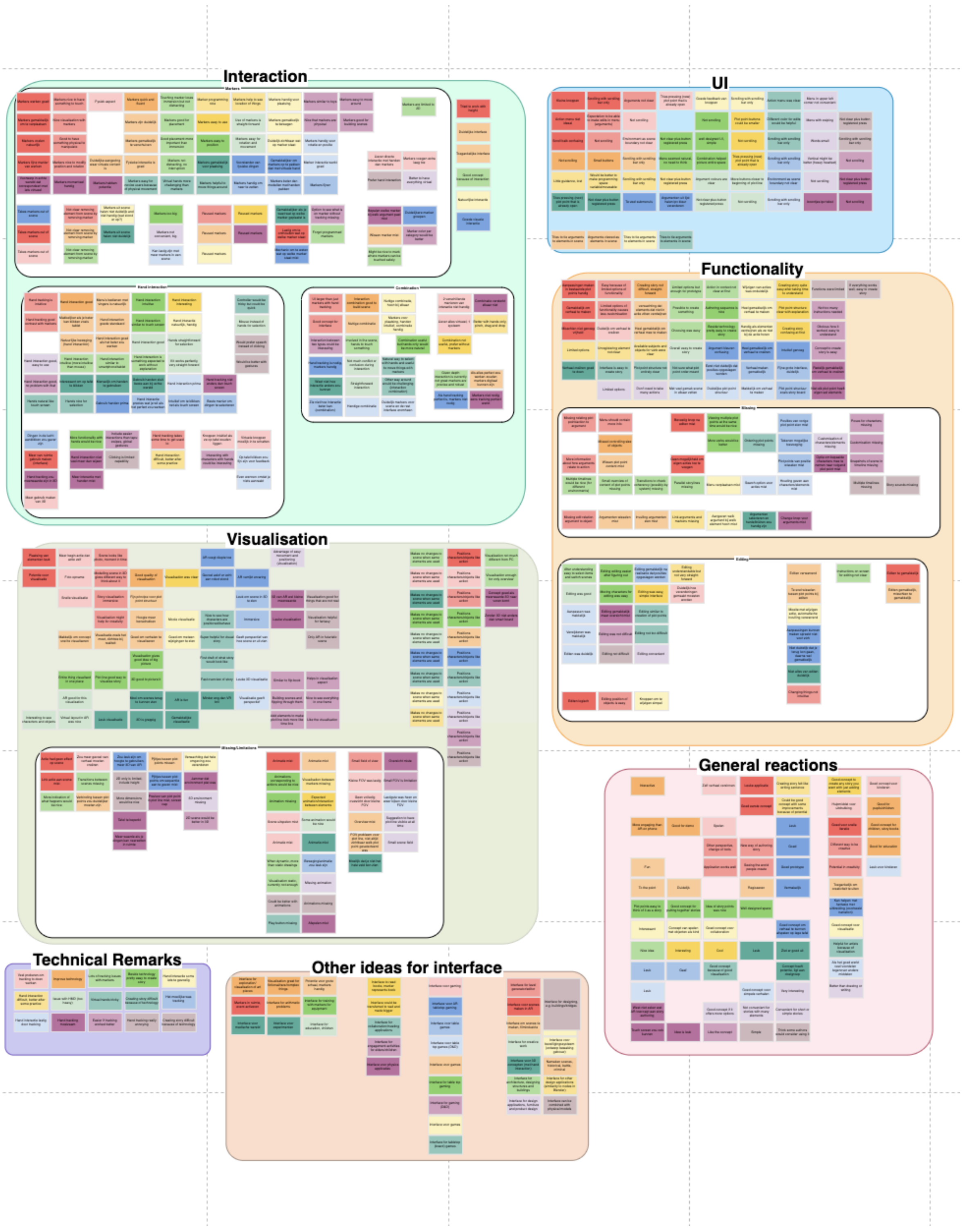


Figure 6.2: An overview of the final affinity diagram containing all clustered notes from the video recording observations and interview answers.

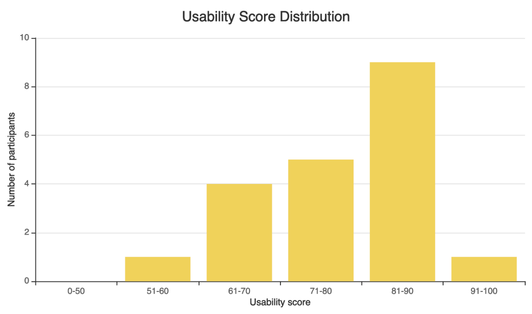


Figure 6.3: Distribution of the usability scores over participants.

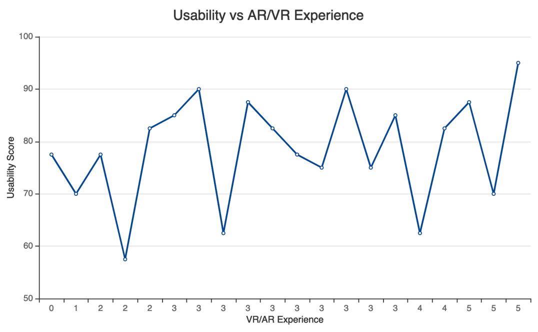


Figure 6.4: Usability scores compared to participant's AR/VR experience.

that was mentioned by 5 people was that there was no way of knowing what was programmed on which marker when a marker was out of the AR device's view or not being tracked. This made people forget what was on each marker. An operation that was unclear to many people was removing an element out of a scene by taking the marker out of the application space. Only 3 people took markers out of the scene without instruction from the observer, and 5 people expressed it was unclear that elements could be removed from the scene by removing the marker.

Specifically for hand interaction, a majority (16/20) explicitly mentioned liking the hand interaction, describing it as intuitive, natural and easy to use. Some compared it with interacting with a touch screen which felt very familiar. Some participants, including the 4 not endorsing the hand interaction, mentioned preferring hand interaction in a 3D space or more complex gestures. Instead of only clicking buttons on the table, they suggested vertically floating menus and gestures like swiping.

Visualisation

Almost all participants (18/20) expressed positive opinions about the AR visualisation. Many mentioned that the 3D aspect of AR is a good way to picture a scene. Some compared the scene visualisation to a snapshot of the current action in the story. It was often mentioned the scenes are an easy way to quickly compose a visualisation of an idea or fantasy.

Even though many endorsed the visualisation, barely half of the participants (11/20) utilised the scene visualisation as intended. This was measured by checking whether participants positioned their characters and objects in a way that made sense related to their chosen actions, e.g. characters facing each other for the action *greet*. In the video footage, it was clear that only 11 people positioned their characters and objects according to the chosen action. All others put each marker on a random location in the scene, often one next to the other. Out of those 9 people, 5 also did not make changes to the scene when authoring a new plot point and no new element needed to be added, even if the elements that were present in the scene and their position made no sense related to the action. However, when the observer mentioned to the participant they could explore the interface and author their own plot points after the tutorial or to go back to authored plot points and edit things, more people would position elements to act out the plot point's action.

Many participants (14/20) reported missing some animations to watch the story play out, some mentioning that it is currently too static or that a connection between the scene and the action was missing. As mentioned above, a framework to implement animations using predicates was designed but was not completed before the user study.

Another limitation that was mentioned was the 2D environment and the 2D nature of the scene in general. Scene elements can only be placed on the surface of the table which some participants found too limiting. While characters and objects are 3D, scene environments are currently 2D images displayed on the table which does not take full advantage of AR visualisation.

The small field of view was also noticed by participants (9/20). They reported not having a proper overview of the application and having to move their head a lot. This was especially viewed as a limitation for the plot line, with participants wanting it to be visible at all times to know which plot point was selected. This issue would probably be mentioned a lot less if the technology was not a limiting factor and the plot line and other interface elements could be projected as intended.

Functionality

When asked about the general task of writing a story using the Story ARtist application, a vast majority (17/20) indicated this to be easy. Some reasons given were the easy and accessible interface, the visualisation of everything in one place, straightforward interaction and the small number of operations required. Some mentioned that the application itself is easy, but the particular technology used made it harder.

Many participants had suggestions about missing functionality but only a few expressed that the current functionality was very limiting (3/20). The missing functionality that was mentioned most often was related to arguments. Simply indicating whether an element was present in the scene that fills the argument was not enough. People wanted to know how the arguments were filled, i.e. which element corresponded to which argument. Elements are automatically assigned to arguments when present in the scene but people wanted the option to change them by pressing a button. Some other suggestions were related to plot points and a connection between them. For example, some participants wanted an overview of the plot point's content in the plot line. Others wanted to see transitions between plot points, see locations of elements in the previous plot point or view multiple plot points at the same time. A few participants were missing the ability to customize things, for example changing poses, colours or sizes of elements.

Opinions about plot point editing were quite divided. A majority (13/20) thought editing was easy but 4 of them mentioned it took some time to understand it. A few participants (5/20) explicitly expressed confusion because the interface did not provide enough information on how editing works, some others thought editing was too easy because everything was saved instantly and there was no button to confirm changes.

Other Interface Use Cases

To assess whether people would find this interface useful for other kinds of applications, the last question of the interview asked for their ideas. Some participants already mentioned other use cases for the interface earlier in the interview as well. The most frequent idea was (tabletop) gaming, which was suggested by 9 participants. Another popular use case was design and creative applications. Although 14 ideas were given by 10 participants, each suggestion was very different. Some examples are game level design, architecture, product design, scene design for films and recreation of historical scenes.

6.4. Discussion

During the development and testing of the Story ARtist application, it became clear that the available technology did not suffice to make the application work smoothly. Even though the setup worked, it was very inconvenient and there was no available way to make all devices work well together. Even after asking participants to focus on the application only, more than half expressed frustrations about the technology during the interview. However, general reactions were mostly positive when asked for some first impressions and throughout the entire interview. It is, of course, important to consider that this type of AR is a novelty for many, which could cause more excitement in participants and therefore more positive reactions.

Because 14 out of 20 participants valued the combination of hand interaction and markers, this can be seen as favorable. As mentioned above, many understand the value of the physical aspect of markers, especially for placement. Participants with significant AR and VR experience, who had tried fully touch-less interfaces before, even expressed the current difficulty of picking up virtual objects using hand interaction only. Without this experience, it is understandable for people to think that hand interaction alone would be more convenient, especially if marker tracking is not working perfectly, which was the case in this prototype. However, hand tracking was not accurate enough in the setup either, and participants had to use a virtual representation of their hands that did not always align perfectly with their physical hands to interact with buttons and menus. This might have given people a better impression of the use of markers.

The Story ARtist application only included tapping and sliding as hand interaction which is why a number of participants mentioned a lack of gestures and interaction in the 3D space. The choice of including only basic hand interaction was done for simplicity and accessibility. In order to include more complex hand interaction while maintaining accessibility, some research would have to be done to find the right balance. Some participants suggested using input speech as well, which is another interaction method that can be researched to add to the combination.

Furthermore, the Story ARtist application can be improved and expanded taking into account the many suggestions made by participants about missing functionality, existing functionality that should be modified and UI adjustments. Because many participants mentioned the lack of animation and since a framework was already designed to include it in the application, this is one of the first improvements that should be added. Something that was unclear to a significant amount of participants was the way arguments were visualised and how they could be changed. This could be one of the first modifications made to the application. Story ARtist can also be expanded in terms of automation to assist the author in writing a story. For example, the data format for verbs can be modified easily to include conditions for consistency checks to make sure the written stories are consistent. Using these consistency checks, the application can also make logical suggestions to the author.

Because a vast majority indicated (i) the application to be a good concept for story authoring in AR and (ii) that creating a story was easy, it can be concluded that there is significant potential in using AR for story authoring with a focus on accessibility. Furthermore, the application received a usability score that is well above average. There was no indication that only the questionnaire answers from people with significant VR/AR experience resulted in a high usability score which is a favorable result for accessibility. Because participants had several ideas about how to use the presented interface for other purposes, there is potential in using AR for other content creation applications.

7 Conclusion

So far, content creation applications have predominantly used conventional methods and devices, often lacking proper visualisation and direct interaction. Augmented Reality has the potential to overcome these limitations. In this thesis, the use of AR for story authoring, a specific case of content creation, was evaluated. The advantages of existing AR interaction types, tangible and touch-less, were combined into a single interface. An AR application was developed with hand gestures as touch-less interaction for selection, and markers as tangible interaction for programming elements and placing them in the scene.

The research question for this thesis asks how AR can be used for story authoring. Because according to most participants, the combination of markers and hand tracking is favorable, it can be said that AR can be used for story authoring by providing an interface that combines tangible and touch-less interactions in a useful way and focuses on simplicity. Results from the user study showed that participants appreciate the physical aspect of the markers and the intuitiveness of the hand interaction. People who are familiar with AR and VR, as well as people without any previous experience, gave the application a usability score well above average which is favorable for accessibility. Furthermore, evaluation results confirmed that 3D visualisation significantly increases the potential of using AR for story authoring. The amount of different ideas expressed by participants to use the designed interface for other creative applications, confirms the high potential in AR for content creation in general.

Future work should use a setup with well-integrated devices, to provide more convenient interaction without the present issues. Other interaction techniques could be investigated, including more complex or different hand interaction and the use of speech as input. Furthermore, the Story ARtist application can be improved primarily by adding animations and modifying the argument system and its functionality. Afterwards, suggestions from the user study regarding missing or inconvenient functionality can be implemented. Automation in the form of suggestions and consistency checks can be added as well to assist the author while writing a story.

A
Resource Files

Action File

```
{
  "Categories": ["Communication", "Contact", "Consumption", "Creation",
                   "Destruction", "Exchange", "Possesion", "Social"],
  "Actions":
  {
        "Communication":
        {
                "Greet":
                {
                        "Arguments": ["Character", "Character"],
                        "Predicates": ["Move(Character1, Character2)",
                                        "Wave(Character1)"]
                },
                "Talk":
                {
                        "Arguments": ["Character", "Character"],
                        "Predicates": ["Move(Character1, Character2)",
                                        "Talk(Character1)"]
                }
        },
        "Contact":
        {
                "Hug":
                {
                        "Arguments": ["Character", "Character"],
                        "Predicates": ["Move(Character1, Character2)",
                                        "Hug(Character1, Character2)"]
                }
        },
        "Consumption":
        {
                "Eat":
                {
                        "Arguments": ["Character", "Prop"],
                        "Predicates": ["Move(Character1, Prop1)" ,
                                        "GrabObject(Character1, Prop1)",
                                        "Eat(Character1, Prop1)"]
                },
                "Drink":
                {
                        "Arguments": ["Character", "Prop"],
                        "Predicates": ["Move(Character1, Prop1)" ,
                                "GrabObject(Character1, Prop1)",
                                "Drink(Character1, Prop1)"]
                }
        },
        "Creation":
        {
                "Make":
                {
                        "Arguments": ["Character", "Prop"],
                        "Predicates": ["MakeObject(Character1, Prop1)",
                                "PlaceObject(Character1, Prop1)"]
                }
        },
        "Destruction":
        {
                "Destroy":
                {
                        "Arguments": ["Character", "Prop"],
                        "Predicates": ["Move(Character1, Prop1)",
                                "DestroyObject(Character1, Prop1)"]
                }
        },
        "Exchange":
        {
                "Give":
                {
                        "Arguments": ["Character", "Character", "Prop"],
                        "Predicates": ["Move(Character1, Prop1)",
                                "GrabObject(Character1, Prop1)",
                                "Move(Character1, Character2)",
                                "GiveObject(Character1, Character2, Prop1)"]
                }
        },
        "Possesion":
        {
                "Take":
                {
                        "Arguments": ["Character", "Prop"],
                        "Predicates": ["Move(Character1, Prop1)",
                                "GrabObject(Character1, Prop1)"]
                },
                "Place":
                {
                        "Arguments": ["Character", "Prop"],
                        "Predicates": ["PlaceObject(Character1, Prop1)"]
                }
        },
        "Social":
        {
                "Meet":
                {
                        "Arguments": ["Character", "Character"],
                                ”Predicates”: [”Move(Character1, Character2)”]
                        },
                        ”Help”:
                        {
                                ”Arguments”: [”Character”, ”Character”],
                                ”Predicates”: [”Move(Character1, Character2)”,
                                                ”Help(Character1, Character2)”]
                        }
                }
        }
}
```

Character File

```
{
        ”Categories”: [”Robots”],
        ”Characters”:
        {
                ”Robots”: [”Robot1”, ”Robot2”]
        }
}
```

Prop (Object) File

```
{
        ”Categories”: [”Consumables”, ”Other”],
        ”Props”:
        {
                ”Consumables”: [”Burger”, ”Oil”],
                ”Other”: [”Drone”]
        }
}
```

Environment File

```
{
        ”Categories”: [”Planets”, ”Space”],
        ”Environments”:
        {
                ”Planets”: [”Mars”, ”Mercury”],
                ”Space”: [”Stars”]
        }
}
```

B Evaluation Materials

B.1. Interview Questions

- What did you think of the application?
- What did you think of the use of the markers?
- What did you think of using your hands for selection?
- Did you find the combination of hand tracking and markers useful? Why (not)?
- How easy or difficult was it to use the interface to create a story? Why?
- How easy or difficult was it to edit your created story?
- Was the plot point structure clear? What would you suggest to improve it?
- What are your thoughts on how the story was visualised?
- Do you think this application is a good concept for story authoring? Why?
- Is there any functionality that you missed?
- Do you think this interface could be used for other AR applications with other goals and domains? If so, can you think of an example?

B.2. Questionnaire

Below are the statements from the updated SUS [28] that participants had to give a score from 1 to 5.

- I think that I would like to use this application frequently
- I found the application unnecessarily complex
- I thought the application was easy to use
- I think that I would need the support of a technical person to be able to use this application
- I found that the various functions in this application were well integrated
- I thought that there was too much inconsistency in this application
- I would imagine that most people would learn to use this application very quickly
- I found the application very awkward to use
- I felt very confident using the application
- I needed to learn a lot of things before I could get going with this product

Bibliography

[1] J. Preece, H. Sharp, and Y. Rogers, *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2015.

[2] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, "Virtual object manipulation on a table-top ar environment," in *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. Ieee, 2000, pp. 111–119.

[3] F. Zhou, H. B.-L. Duh, and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ismar," in *Proceedings of the 7th IEEE/ACM international symposium on mixed and augmented reality*. IEEE Computer Society, 2008, pp. 193–202.

[4] I. Poupyrev, D. S. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani, "Developing a generic augmented-reality interface," *Computer*, vol. 35, no. 3, pp. 44–50, 2002.

[5] V. T. Phan and S. Y. Choo, "Interior design in augmented reality environment," *International Journal of Computer Applications*, vol. 5, no. 5, pp. 16–21, 2010.

[6] M. Kapadia, J. Falk, F. Zünd, M. Marti, R. W. Sumner, and M. Gross, "Computer-assisted authoring of interactive narratives," in *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*. ACM, 2015, pp. 85–92.

[7] Y. Shen, S. Ong, and A. Y. Nee, "Augmented reality for collaborative product design and development," *Design studies*, vol. 31, no. 2, pp. 118–145, 2010.

[8] Z. Lv, A. Halawani, S. Feng, S. Ur Réhman, and H. Li, "Touch-less interactive augmented reality game on vision-based wearable device," *Personal and Ubiquitous Computing*, vol. 19, no. 3-4, pp. 551–567, 2015.

[9] S. Kim and A. K. Dey, "Ar interfacing with prototype 3d applications based on user-centered interactivity," *Computer-Aided Design*, vol. 42, no. 5, pp. 373–386, 2010.

[10] H. Benko, E. W. Ishak, and S. Feiner, "Collaborative mixed reality visualization of an archaeological excavation," in *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2004, pp. 132–140.

[11] L. X. Ng, S. Oon, S. K. Ong, and A. Y. Nee, "Garde: a gesture-based augmented reality design evaluation system," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 5, no. 2, p. 85, 2011.

[12] S. Malik, C. McDonald, and G. Roth, "Hand tracking for interactive pattern-based augmented reality," in *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2002, p. 117.

[13] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 3633–3642.

[14] J. Singha, A. Roy, and R. H. Laskar, "Dynamic hand gesture recognition using vision-based approach for human–computer interaction," *Neural Computing and Applications*, vol. 29, no. 4, pp. 1129–1141, 2018.

[15] S. Göbel, L. Salvatore, and R. Konrad, "Storytec: A digital storytelling platform for the authoring and experiencing of interactive and non-linear stories," in *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*. Ieee, 2008, pp. 103–110.

[16] B. Medler and B. Magerko, "Scribe: A tool for authoring event driven interactive drama," in *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*. Springer, 2006, pp. 139–150.

[17] J. J. LaViola Jr, E. Kruijff, R. P. McMahan, D. Bowman, and I. P. Poupyrev, *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.

[18] "Unity," Unity Technologies, https://unity.com/.

[19] "Unity assets for leap motion orion beta," Ultraleap, https://developer.leapmotion.com/unity5436356.

[20] "Vuforia engine," PTC, 2020, https://www.ptc.com/en/products/augmented-reality/vuforia.

[21] "Json .net for unity," parentElement, https://www.parentelement.com/assets/json_net_unity.

[22] K. K. Schuler, "Verbnet: A broad-coverage, comprehensive verb lexicon," 2005.

[23] B. Kybartas and R. Bidarra, "A semantic foundation for mixed-initiative computational storytelling," in *International Conference on Interactive Digital Storytelling*. Springer, 2015, pp. 162–169.

[24] ——, "A survey on story generation techniques for authoring computational narratives," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 3, pp. 239–253, 2016.

[25] S. W. Brown, J. Pustejovsky, A. Zaenen, and M. Palmer, "Integrating generative lexicon event structures into verbnet," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[26] S. W. Brown, J. Bonn, J. Gung, A. Zaenen, J. Pustejovsky, and M. Palmer, "Verbnet representations: Subevent semantics for transfer verbs," in *Proceedings of the First International Workshop on Designing Meaning Representations*, 2019, pp. 154–163.

[27] J. Brooke, "Sus: a "quick and dirty'usability," *Usability evaluation in industry*, p. 189, 1996.

[28] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[29] A. Lucero, "Using affinity diagrams to evaluate interactive prototypes," in *IFIP Conference on Human-Computer Interaction*. Springer, 2015, pp. 231–248.

[30] H. Beyer and K. Holtzblatt, "Contextual design," *interactions*, vol. 6, no. 1, pp. 32–42, 1999.