AR MUSE: Designing and Implementing a Solution for Accessible Augmented Reality Exhibition

by
Qianlin Duanmu
Tian Dai
Yingcheng Cai
Julian Herman

A Thesis Project
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Interactive Media and Game Development

April 2023

APPROVED:

Joshua Rosenstock, Project Advisor, WPI

Charles Roberts, Committee, WPI

Gillian Smith, Committee, WPI

Abstract

The following thesis report explores the development, design, and usefulness of MUSE, an Android application that facilitates the integration of 3D models with QR codes to enable augmented reality (AR) visualization. This study analyzes the current status of AR and its applications in various fields and how our designed application could be best utilized. The paper discusses the technical specifications and functionality of the MUSE application, including its interconnected nature with the game engine Unity and the importation, exportation, and visualization processes of 3D models. The study concludes by evaluating the potential versatility of MUSE in different scenarios, such as education, marketing, and entertainment. The results show that MUSE provides an accessible, user-friendly, and cost-effective solution for displaying 3D models in AR. Through this, we hope to make AR a more accessible and viable presentation option for individuals who use 3D models the world over.

Acknowledgments

Our team would thank everyone that made this project possible.
We thank our advisors Joshua Rosenstock and Charles Roberts for being able to meet with our team on a weekly basis and provide us suggestions and directions for the project. Individually, we thank Joshua Rosenstock for explaining AR museum, and helping provide suggestions and directions on the fundamentals of this project, and ideas on how to improve and simplify the progress. We thank Charles Roberts for assisting us in project organization, and management, and we also appreciate correcting our mistakes.
We thank the WPI Interactive Media and Game Development department for providing us server space at the 2022 Alpha Fest to conduct a very early stage test, which strengthened our goal. We also thank Max Chen for inspiring us in the form of an AR museum.

Authorship

All team members contributed to the creation of this project report.

Table of Contents

Abstract **1**
Acknowledgments **2**
Authorship **3**
Table of Contents **4**
1. Introduction **6**
1.1. Context 7
1.2. Project Motivation 7
1.3. AR vs Traditional & Web Museum Exhibitions 8
1.4. Costs with Curation of an Augmented Reality Museum 8
2. Background and Related Work **10**
2.1. Background 10
2.1.1. The State of AR Development Tools 11
2.2. Related Work 12
3. AR MUSE Design and Implementation **14**
3.1. Systematic Design of AR MUSE 14
3.1.1. AR MUSE Project Design Goal 14
3.1.2. Design of AR MUSE 15
3.1.2.1. General Design 15
3.1.2.2. Creator Platform Design 17
3.1.2.3. Audience Platform Design 18
3.1.2.4. Server Platform Design 18
3.1.2.5. Connecting Creators and Exhibition Audiences 19
3.2. AR Muse Components Design 20
3.2.1. AR MUSE Curator App (Legacy) 20
3.2.2. AR MUSE Unity Toolkit 21
3.2.3. AR MUSE Viewer App 22
3.3. Interaction and Experience Design 24
3.3.1. Interacts with Space: QR Code 24
3.3.2. Interacts with Virtual Exhibit: Viewer App 25
3.3.3. Interacts with the Mobile Viewer App: Viewer App UI Design 27
3.3.4. Interacts with the Unity Toolkit: Unity Toolkit Design 29
3.4. AR MUSE Implementation 30
3.4.1. Unity Toolkit Implementation 31

3.4.2. Viewer App Implementation 33
4. Evaluation 36
4.1. Early Development Test 36
4.2. Evaluation Design 36
4.2.1. Pre-Test Survey 36
4.2.2. Toolkit Test 37
4.2.3. Viewer App Test 37
4.2.4. Post-Test Survey 37
4.2.4.1. Quantitative questions 37
4.2.4.2. Qualitative Questions 38
4.3. Evaluation Result 38
4.3.1. Quantitative Questions Result (System Usability Scale) 38
4.3.2. Qualitative Questions Result 40
4.4. Discussion 41
5. Post-mortem 42
5.1. Achievements and Lessons Learned 42
5.2. Future Work 45
5.3. Conclusion 45
Bibliography 46
Appendix: Playtesting Questionnaire 48
1. Background Survey 48
2. System Usability Scale 51
3. Interview Questions 56
Appendix: AR MUSE Unity Toolkit Test Task 62
Appendix: Detailed Information of AR MUSE 67
1. Curator App 67
2. Unity Toolkit 68
3. Viewer App 71

1. Introduction

With our app, AR MUSE, 3D artists now have the capability to exhibit their creative works to visitors easier than ever before. By pointing your cell phone camera at a QR code, potential viewers can see a person's 3D artwork in an intuitive and tangible manner akin to appreciating a display piece in a museum. Having visitors take a look at and enjoy a piece of digital art without having to navigate any distracting user interfaces will in turn, make the space between user and viewer less complicated than ever.

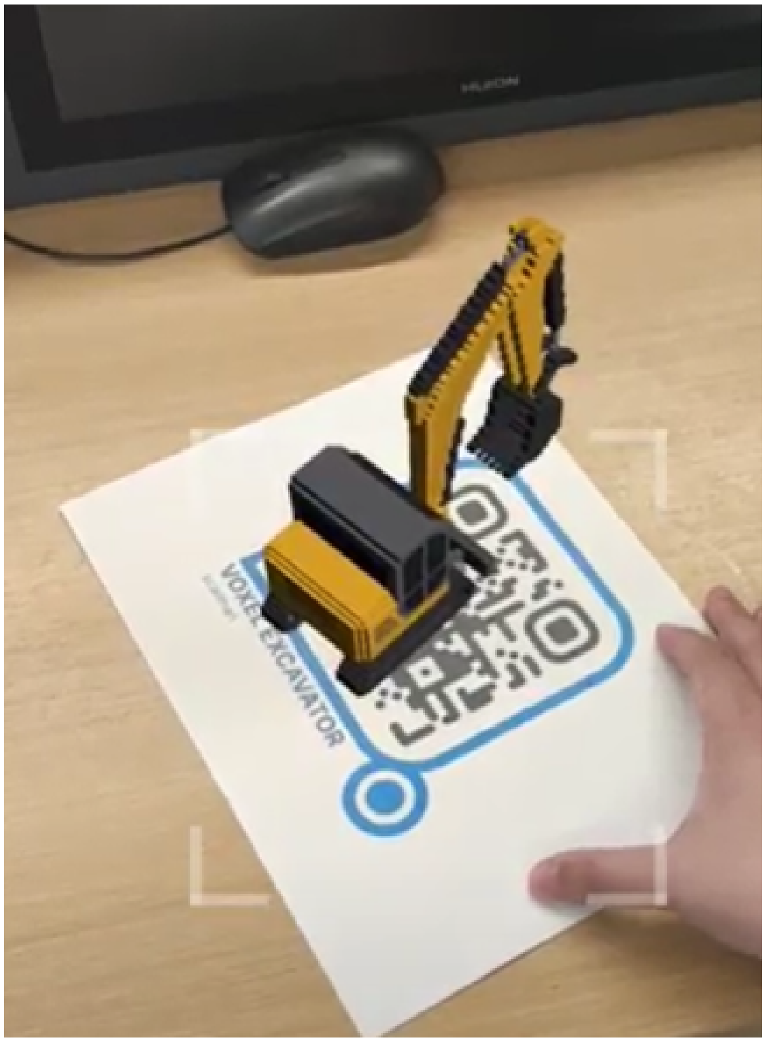


Figure 1: MUSE AR Viewer App Presenting a 3D Model through a Phone Camera

1.1. Context

By its nature, the methodology for viewing and appreciating 3D design is inexorably tied to viewing it through a screen. And through the pandemic, art appreciation is more important than ever as a vital way to share one's creative endeavors with those far away. To view the majority of virtual art, however, potential viewers have to navigate obtrusive digital interfaces. Concurrently, digital artists who want to share their creative works with those unfamiliar with navigating these interfaces often have difficulties compiling and exhibiting their work so that others can see it. This barrier to access increases the bar with regards to digital art sharing and appreciation, making it more difficult to organize and share one's digital work in a non-digital environment.

Our research lowers that barrier of entry for virtual art appreciation through the use of AR technology. To standardize, personalize, and consolidate different aspects of digital art presentation by giving potential viewers a tangible method of artistic admiration. And to give users an easy way to organize and show their work to those viewers through an immersive and customizable museum experience.

1.2. Project Motivation

In Kan's article he proposed the idea of popping up an item model while scanning the item's barcode in store[1]. Kan's paper gives us an idea of using QR codes as a more general medium for AR artworks. First, the QR code represents a plane. The second is that the QR code itself has its own directionality. Last but not least is the information storage property that comes with the QR code itself [2].

We ended up with the solution: store the file in the cloud server and use the orientation property of the QR code as the coordinate reference for generating the model. At the same time, the QR code also stores the link to access the cloud with the information storage attribute. That will make our app lightweight and highly generalizable.

1.3. AR vs Traditional & Web Museum Exhibitions

Traditional museum exhibits offer visitors a physical space with curated objects on display. Visitors can view and learn about the objects in person, but the experience is largely passive. Website exhibits however provide visitors with a virtual space to explore collections and learn about objects. These exhibits often offer interactive features such as zooming in on images or clicking on links for more information. However, the experience is still largely two-dimensional and detached from the physical objects themselves.

AR museum exhibits offer a hybrid and tangible experience that combines the best of both traditional and online exhibits. With mobile AR technology, visitors can view objects in a physical space while also receiving additional information and interactive features through their mobile devices. Ideally, this in turn creates a more immersive and engaging experience that can enhance visitors' understanding and appreciation of the collections. AR museum exhibits also have the advantage of being customizable and adaptable. Museums can update and change exhibits more easily with AR, allowing them to showcase different collections or themes without the need for extensive physical renovations or layout changes.

However, there are also potential drawbacks to AR museum exhibits. The key issue is that some visitors may find the technology distracting or difficult to use, which could detract from the overall experience and be fatal to an exhibit's sense of immersion.

1.4. Costs with Curation of an Augmented Reality Museum

Curating an Augmented Reality museum is inherently different from curating a traditional, tangible museum, with different considerations and deliberations taking place. As such, the process can be a costly endeavor for those looking into it, but once completed and fully realized, the potential benefits can outweigh the initial investment hesitancy. One of the primary expenses involved in curating an AR-based museum is the creation and implementation of AR content specifically appropriate for the exhibition. This content can range from the development of 3D models and spaces, animations, and interactive elements such as the user interface and user experience that are overlaid onto the real-world environment. Each of these elements requires not only the efforts put in by the artists but also purposeful curation and design to achieve the desired immersive effect on the viewer. Another significant cost associated with curating an AR museum is the technology required to deliver the experience. Augmented reality can only be viewed through a digital screen, and as such, obtaining the necessary hardware can at times be costly. Appropriate hardware, such as smartphones, tablets, or dedicated AR headsets, in addition to the platform's software and development and curation tools would be necessary for viewing and curating an augmented reality environment. The cost of these components can vary widely depending on the chosen technology and the number of devices needed to deliver the experience, with in 2022 the average price of a consumer tablet being around $178 [3]. Finally, the physical space of the museum itself can also impact curation. In order to create an AR experience, the physical space must be carefully mapped to ensure the AR content fits as seamlessly and unobtrusively into the environment as possible. This can require additional retrofitting of the museum layout, which can further add to the overall cost of the project. While the expenses may seem exorbitant at first, the benefits of the additional control over the exhibit pieces and the immersive interactivity, give AR museums an experience that cannot be found anywhere else and is worth the cost.

The cost of curation of a digital museum can very easily balloon and inflate, with expensive viewing technology on both the software and hardware front as well as trying to incorporate the digital pieces into a tangible environment. The solutions to ameliorate both of these costs would be to have a piece of AR viewing software readily available for free on any phone, and to have easily generated and accessible digital markers (QR codes) that can be placed on any section or on any surface of any given environment. With these advents, curation of an augmented reality museum will be easier than ever with the task being much more available and affordable.

2. Background and Related Work

2.1. Background

Augmented reality (AR) is a technology that overlays digital information and images onto the real world, creating a mixed reality experience. AR typically involves the use of a camera-equipped device, such as a smartphone or tablet, which displays a live video feed of the user's surroundings. Digital elements are then superimposed onto this live video, making it appear as if the digital objects are part of the real-world.

AR is commonly used today in a variety of applications, including:

1) Gaming: AR has been used in a number of popular games, such as “Pokemon Go”[4] and “Ingress”[5] , where players use their mobile devices to explore real-world environments and interact with virtual objects and characters.
2) Advertising and marketing: AR can be used to create interactive product demonstrations and advertising campaigns, such as Burberrys Olympia pop up ad campaign[6], allowing consumers to engage with products and brands in a more immersive and attention grabbing way.
3) Education and Training: AR can be used to create interactive educational experiences, such as virtual field trips, anatomy lessons, and historical recreations. We already see AR based teaching tools become more prominent in classroom spaces, such as the educational astrology app "Skyview”[7] and the exploratory biology app “Froggipedia”[8].
4) Architecture and Design: AR can be used to create virtual models of buildings and other structures, allowing architects and designers to visualize and manipulate their designs in real-time. Morpholios “AR Sketchwalk”[9] is a tool that allows architects to see digital layouts and floor plans in a real-world setting to give them a sense of size and space prior to construction.
5) Retail: AR can be used to create virtual try-on experiences, allowing customers to see how products such as clothing or makeup would look on them before making a purchase. The furniture store IKEA has famously already used this technological prospect as the

basis for their AR application "IKEA Place"[10] which allows users to see how a potential product would fit in their living space.

2.1.1. The State of AR Development Tools

The following are several pre-existing pieces of computer software that support augmented reality and are readily available on the market.

- "Vuforia"[11] is an AR software platform that allows developers to create AR applications for mobile devices, tablets, and smart glasses. It offers robust customization features such as image recognition and object tracking. A limited version is available for free, but the paid edition with all the features starts at a steep $99 a month.
- "ARKit"[12] is a piece of software with a suite of features and an intuitive interface that allows individuals to make their own augmented reality experiences for free. However, the disadvantage is its limited availability, as it is only available on Apple smart devices, limiting its availability through its exclusivity.
- "AR Foundation"[13] is a plugin for the popular game engine Unity which gives users tools for designing compatible AR programs across a wide variety of platforms, which can help to save time on integration and program porting. However despite the versatility of the program, being intentionally designed for not one but many different platforms leaves overall performance uneven across the board.
- "AR Studio"[14] is a web based AR application building platform available exclusively for Mac that allows designers to create and build their own Augmented Reality programs to be used in any available internet browser. However it is limited by requiring a monthly subscription to access systems creation features.

Unity and Vuforia are both available on most platforms, but Vuforia's high barrier of entry makes it an unfavorable choice. Unity, by comparison, is much more customizable and available, and with an entire bevy of documentation behind it, Unity is a clear choice for augmented reality design and usage when it comes to AR application and technologies. For this reason it was appropriate to use Unity to create and design our own AR program to act as an easier , more accessible version of the higher budgeted higher affordance versions available on the current market.

2.2. Related Work

There are already some successful AR Museum App examples, like ReBlink[15], created by digital artist Alex Mayhew (figure 2), and Smartify[16], which was used for The National Gallery in London 2021 (figure 3). ReBlink was using imaging tracking, which identifies a painting then renders the animation of that painting upon that painting. Smartify used QR-code-like images on the wall as a marker to show a specific painting.

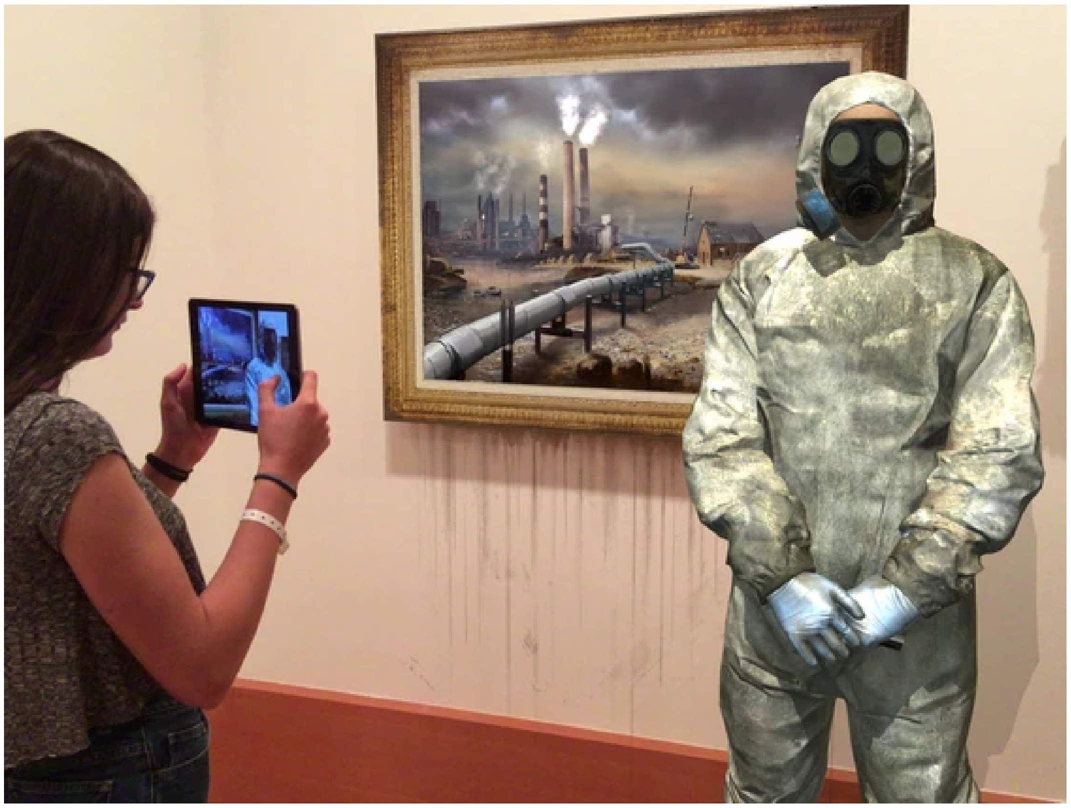


Figure 2: An example of the Augmented Reality museum ReBlink

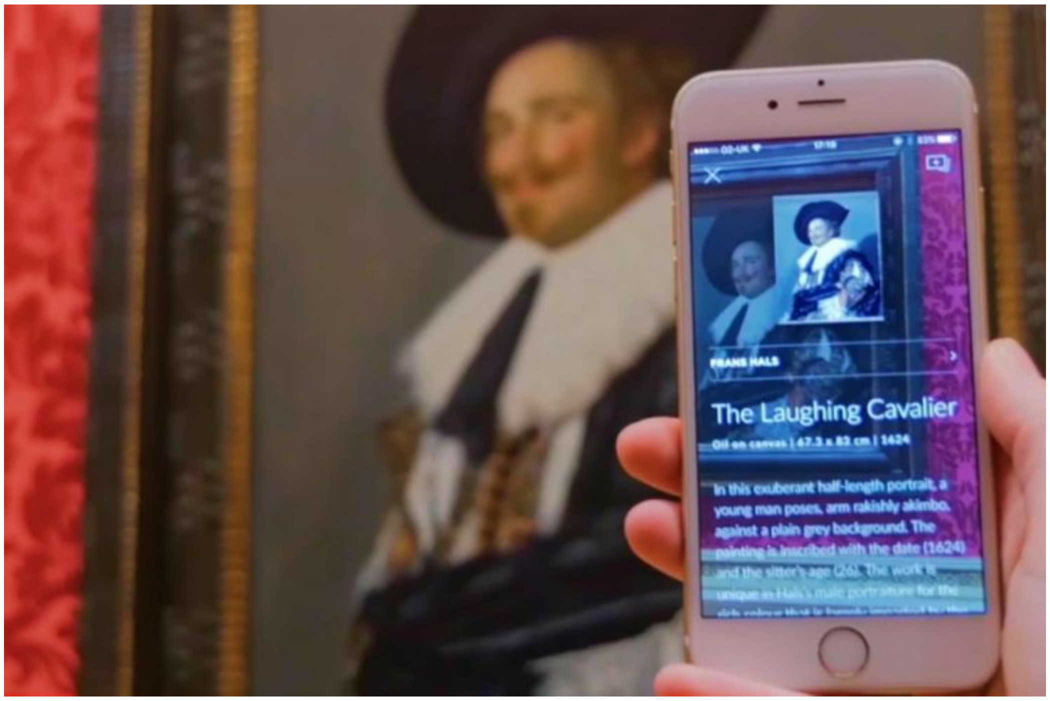

FRANS HALS
The Laughing Cavalier
Oil on canvas | 67.3 x 83 cm | 1624
In this exuberant half-length portrait, a young man poses, arm rakishly akimbo, against a plain grey background. The painting is inscribed with the date (1624) and the sitter's age (26). The work is unique in Hals's male portraiture for the

Figure 3: An example of the Augmented Reality museum Smartify

In terms of successes and failures, ReBlink received widespread acclaim for its innovative approach and use of facial recognition technology. However, its specific brand of digital art remains a costly example both in terms of capital and artistic creation, having to integrate both digital and real-world artifacts for all of its pieces. On the other hand, Smartify has been successful in providing accessible and informative museum experiences, but it has faced criticism for its reliance on smartphones and the potential distractions they create.

Despite the challenges and limitations, AR-based museum applications offer several benefits over traditional exhibits. These applications can provide an interactive and immersive experience for visitors, allowing them to engage with artworks in new and innovative ways. AR-based applications can also provide detailed and contextual information, enhancing visitors' understanding and appreciation of the artwork.

Both of these two Apps and most AR Apps today, have in common is that they require the user to download all the model files to the local device in advance. We recognize this becomes one of the hindrances that limits the dissemination of AR artworks. When viewers attend to enjoy these museum works, they either use the equipment provided by the Museum that already has preloaded everything or they need to download an additional App that contains all the model files.

And our system tries to solve this problem of a steep/costly barrier of entry and a lack of user customization and expression. Users do not need to download all the content at the beginning, but download what they need during the process when they are browsing the museum.

3. AR MUSE Design and Implementation

3.1. Systematic Design of AR MUSE

3.1.1. AR MUSE Project Design Goal

After doing background research on AR applications and relating to our own AR-related development experience, the team discovered that in order to present and share their models, individuals must overcome several technical barriers, such as designing software and coding, operating in Unity editor, and packing and publishing applications to mobile platforms.

Table 1: AR Application Background

	Engine Plug-in (AR Foundation)	Web AR (AR studio)	AR Mobile App (Reality Composer)
Technical Skill	Software Engineer	Entry Programmer	None Required
Customized Model	YES	YES	NO
Animation Support	YES	NO(Mostly)	NO
Organize Exhibition	YES	NO	NO
Open-source	YES	NO	NO
Shareable	NO	NO	NO

While the current affordance may be acceptable for trained professionals, it still remains inaccessible to general designers, artists, and educators. Therefore, the goal of the AR MUSE project is to reduce, and ideally eliminate, the technical barriers associated with sharing, presenting, and organizing AR exhibits, making an integrated solution that is more accessible to everyone.

The system will have two separate sides: Creator side and Viewer side. For the creator side, we originally designed two different Apps for users with different technical skill levels. The Curator App is designed as an independent client that can package, adjust and send assets. The Curator App is aimed towards general designers, artists, and educators who have little to no technical skills in Unity, software design, or coding. In order to reduce the burden on these users, the Curator App has a simple and basic user interface and is only capable of performing very basic 3D model adjustments. The AR MUSE Unity Toolkit is intended for artists and designers who are already familiar with the Unity editor and want to use advanced editing and functions. It operates within the Unity editor, so the experienced users can take advantage of the advanced model-editing, animation, and state machine components in Unity editor while using our packing scripts to deploy their work to the Viewer App. The Viewer App is a user client for mobile platforms that allows users to download packages from either the Curator App or Unity Toolkit and view them in an AR environment via QR codes. Its targeted users are exhibition audiences, so the application needs to be clear and simple. Also, the Viewer App needs to be highly maintainable because there is a lot of update work.

3.1.2. Design of AR MUSE

3.1.2.1. General Design

We chose to create AR MUSE solution by using Unity Engine because it offers powerful edit platforms, and allows artists "what you see is what you get". Also, to ensure maximum compatibility on different devices, Unity's official AR Foundation package is the most important development tool we decide to use. We finalized these three Apps to develop:

- A Curator App: which is a desktop application designed for creators with very limited knowledge of operating with Unity Engine. It provides a friendly and easy to understand

GUI to allow creators to modify and pack models, as well as a one-click upload function, but also lacks powerful editing tools.

- A Unity Toolkit, which is a Unity package for creators to import and make customizations to their models, this toolkit will also help artists to pack models into asset bundles, and generate the config file without any coding.
- A Viewer App, which is an Android mobile application developed by Unity, allows exhibition audiences to scan QR codes and inspect these models through their phone's camera.

In order to make exhibition audiences (users of the Viewer App) able to fetch models remotely, a QR code generator and a cloud server are the external parts of the system. A cloud server will be a bucket to store these models and the config file, and the QR code generator is used to generate some fascinating QR codes.

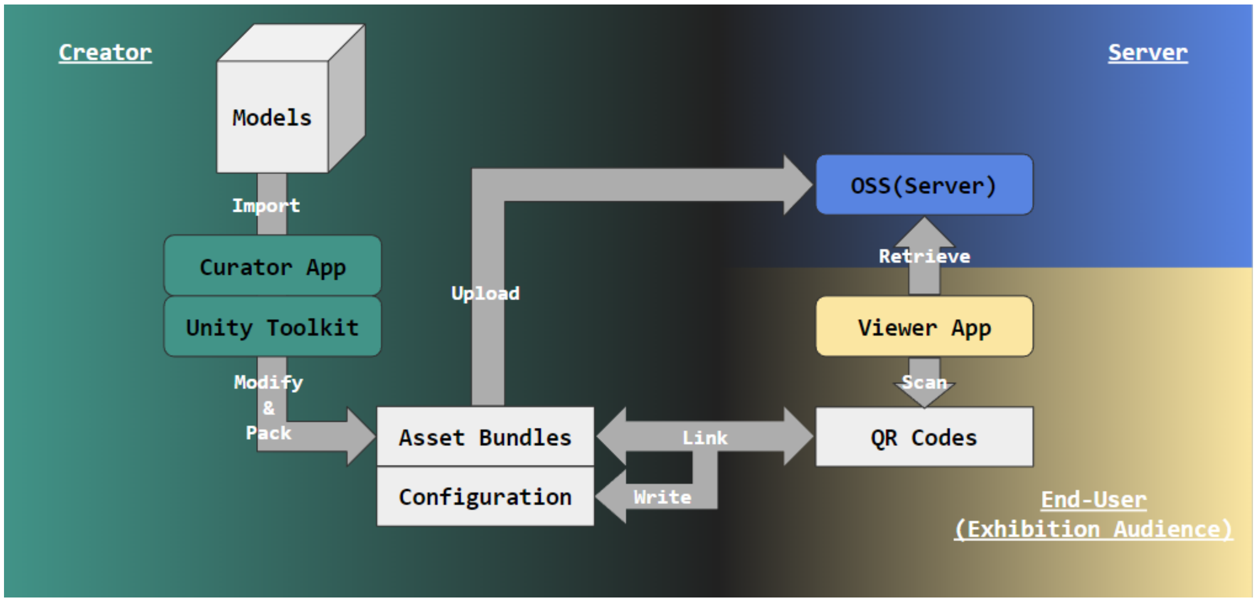


Figure 4: Workflow of AR MUSE solution

After introducing every part of this system, it is time to combine them together. The creator needs external 3D modeling software like Blender, or 3D MAX to produce some Unity Engine-ready models. They also need to import the AR MUSE Unity Toolkit into an empty project, and drag these models into Unity. Then they need to use the Asset Bundle Packer to build asset bundles for their model, and Config Generator to create a config file. The two tools are included in the AR MUSE Unity Toolkit. After getting these asset bundles and a config file, they have to upload them to an Object Storage Server (OSS) to store them, and generate entry QR code, as well as QR codes for each single model based on the link of these files. Now, an AR museum is ready for receiving exhibition audiences. They only need to download the Viewer App before entering the museum, after they scan the entry QR code, all the other models are accessible to them through their phone's camera when they scan a single model's binded QR code.

Detailed steps of how this solution works will be explained in chapter 3.1.2.5**.**

3.1.2.2. Creator Platform Design

The whole project idea originated from our desire to find a solution to lower the barriers to creating AR museums. AR museums still have commonality with traditional physical museums. We initially only considered providing a unique QR code for each exhibit. But after our analysis of museums, we concluded that in a museum, there can be multiple exhibition halls, and then there can be multiple exhibits in one exhibition hall. We treat the two as a superior-subordinate relationship. Therefore, in the finalized tool, users need to fill in two forms, one for exhibition hall information and one list for all exhibits. After exporting, the user will get an entry QR code, as well as the QR code of each exhibit.

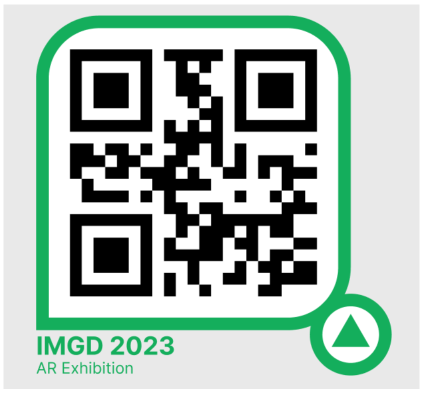


Figure 5: Entry Code

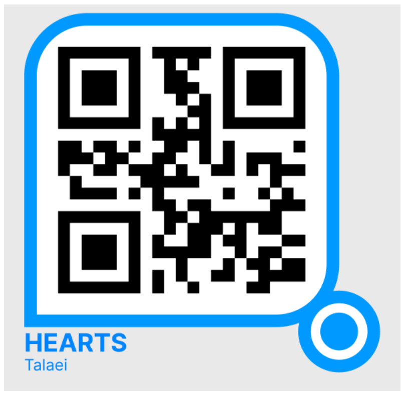


Figure 6: Individual Code

The entrance QR code is like the flier that tourists get at the entrance when they visit a real museum, containing exhibition information such as the name and introduction of the exhibition hall, and a list of exhibits. In addition to retaining the sense of ritual of entering the museum, the existence of the QR code at the entrance is also to facilitate the management of exhibits. When the number of exhibits reaches a certain level, it will be more clear to have a list than to manage each exhibit individually.

Individual QR codes link to individual artwork. Each individual QR code is like a booth in the museum, figure 1 perfectly shows this. There is only one model on each booth. After scanning the QR code at the entrance and getting the list, visitors still need to visit each individual booth to view the works in more detail.

3.1.2.3. Audience Platform Design

Users will need to scan the entry QR code before they enter the exhibition. It's like how visitors grab a flier before they enter the real-world museum, so they can have a general idea of the exhibition and see the exhibit list.

After scanning the entry QR code. Users can now scan any other exhibit QR code that belongs to this exhibition. If they do not first scan the entry code, the viewer app won't detect the exhibit in the exhibition. They need to "enter" the exhibition first.

When users have enough with the current exhibition, users need to "exit" the exhibition hall by selecting the "exit" options in the upper left corner of the screen. So they can join another exhibition.

3.1.2.4. Server Platform Design

Server plays an important role in AR MUSE solution, that it stores all the files, and significantly affects user experience when they try to download. After considering all the server technologies, we believe the Object Storage Server (OSS) is the best choice for several reasons:

- Simple to manage and operate: OSS services usually come with a GUI panel, and most of them are user-friendly even for beginners.
- Enough choices: There are lots of companies offering OSS services, including Amazon, Microsoft, and Alibaba.
- Low price: When comparing renting a fully functional server, renting an OSS is much cheaper. Our team rented an OSS provided by Alibaba Cloud for playtesting, the monthly fee is less than 10 USD per month.

- DDOS protection: DDOS occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers. And most of the providers have a basic DDOS protection, which prevents potential server down when there is a large amount of audience trying to download models from the server.

3.1.2.5. Connecting Creators and Exhibition Audiences

After explaining each component of AR MUSE, in this section, we will briefly discuss how each component is connected to others, and how this solution works.

1) This solution starts with models, as a curator of an exhibit, models need to be imported to AR MUSE Unity Toolkit manually, theoretically any model file format Unity supports could work on mobile devices, but when considering exhibition audiences need to download models from server and relatively poor performance of mobile phones compared with desktop computers, an obj file with reasonable triangles is highly recommended[17].
2) After importing the models, the creator can make some customizations to the models, for example, do some transform modifications, apply an animation clip, or add particle effects. If necessary, the curator can bind additional scripts to models in order to achieve some fascinating effects or make models interactable.
3) Once finished customizations, AR MUSE Unity Toolkit provides an Asset Bundle Packer tool to help curator pack these models into asset bundles with simple drags and clicks.
4) The next step is to set museum information, along with all the models packed previously. Creators can use the Config Generator offered by AR MUSE Unity Toolkit to set museum and all the models' information, for example, title, author, and description. Also, the Load Config function provides curator ability to manage and update existing exhibition's config.
5) When a creator finishes all the steps above, he can upload these packed models and config files to the server, and prepare some QR codes, which will be the identity of the models. In detail, the entry QR code's content must be the direct link to the config file uploaded to the server, and for the packed models, their QR codes' contents are the filename written in the config file.
6) When in an actual exhibition, the exhibition audience will use our Viewer App to scan

these QR codes, which are linked to asset bundles previously, then they can load these amazing models locally on their phones.

3.2. AR Muse Components Design

3.2.1. AR MUSE Curator App (Legacy)

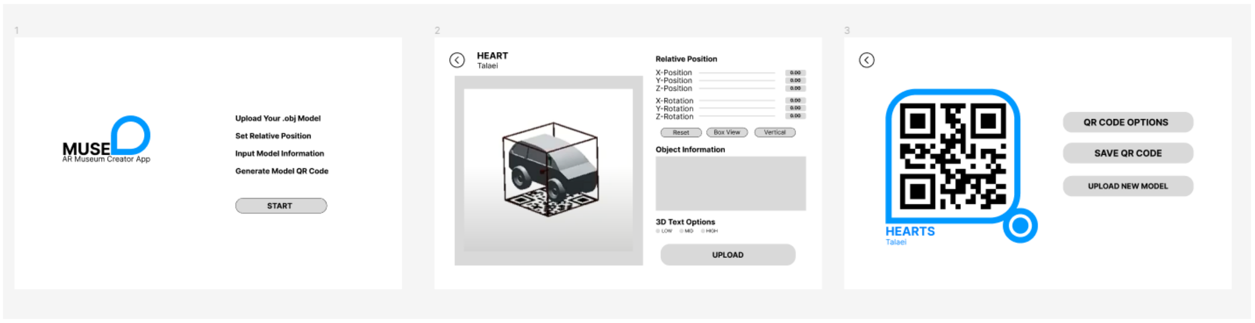


Figure 7: General look of Curator App

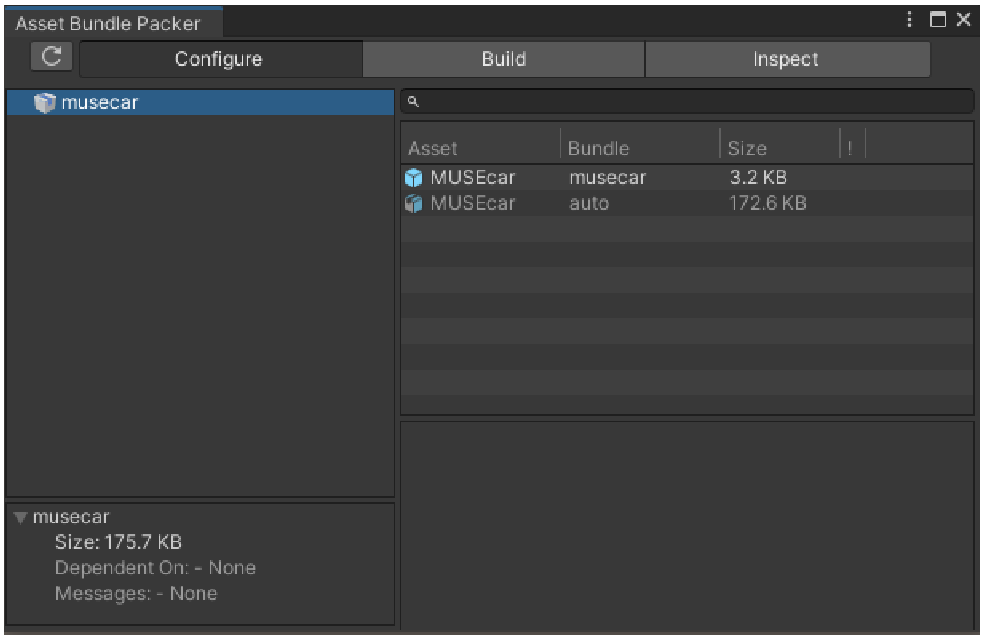
In our initial plan, we had a separate curator app intended for users with limited technical skills. It was designed to be an independent client that is very easy to use, with an easy-to-understand UI and every basic feature. In exchange, the app can only make limited adjustments to the model. In order to achieve this goal, the software also automatically adjusts the imported model files to the appropriate size to fit in the QR code. And the toolkit, in the initial plan, only included scripts we created during the entire development process, like packaging models, and functions such as QRcode detection and model lifetime setting. We were hoping to share those scripts with Unity experts who want to create their own art exhibit. Therefore, the toolkit does not have any UI or interactive interface in its design, and we expect users of the toolkit to know what they are doing and what they are using.

But in the early testing, we found out that there are some misunderstandings between us and creators. We were thinking about how to simplify the system, but the result is that the simplified model editing process is actually not necessary for those who are interested in creating and sharing 3D arts. People don't like the limited editing function. They want to have more freedom so they can show their creativity. Especially the automatic resizing model, and the limited editing features, many people have expressed dissatisfaction with these features that we think are convenient for them. People don't want their creativity to be limited by a creative tool. What they need is a simplified system to help with deploying their models and sharing with the audience.

As a result, we ended up combining the original Curator app with the original Toolkit, creating the Toolkit that was ultimately shipped. We inherited the coordinate reference and upload functions into the Toolkit, which now has a user interface.

3.2.2. AR MUSE Unity Toolkit

Unity Engine provides a powerful platform for artists to modify these models, from basic transformation to binding additional scripts to add features, and it is great to have infinite creativity and possibility for artists to innovate. So we decided to take advantage of this, and combine the requirements of 3D model editing, packing and deploying, so we designed this AR MUSE Unity Toolkit with two tools: Asset Bundle Packer and Config Generator.



Asset	Bundle	Size	!
MUSEcar	musecar	3.2 KB	
MUSEcar	auto	172.6 KB	

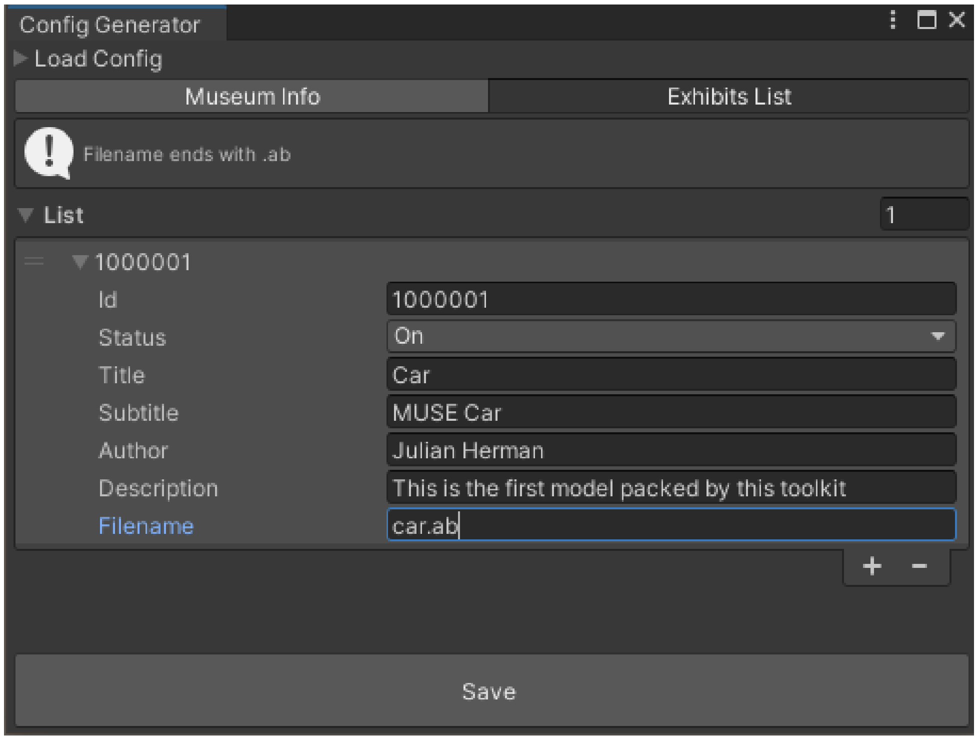
musecar
Size: 175.7 KB
Dependent On: - None
Messages: - None

Figure 8: Asset Bundle Packer Window

An asset bundle is an archive file that contains platform-specific non-code assets (such as

Models, Textures, Prefabs, Audio clips, and even entire Scenes) that Unity can load at run time. In other words, an asset bundle is a combination of model assets, which allows creators to pack whatever they want

Asset Bundle Packer provides functionality that enables users to configure and inspect packed models. The Build function provides users a simple way to pack their models into asset bundles.



Filename ends with .ab

List 1

1000001

Id 1000001
Status On
Title Car
Subtitle MUSE Car
Author Julian Herman
Description This is the first model packed by this toolkit
Filename car.ab

Figure 9: Config Generator Window

Config Generator is especially designed for the requirement of high maintainability, that by re-uploading config files, the curator or owner of the AR museum can easily manage all the exhibits on display, in other words, this config design implements hot-fix.

3.2.3. AR MUSE Viewer App

According to our research on existing AR museums, most of the museums use compatible or specific apps for inspecting the models. Some of them store all the models offline, which is reasonable, because when the museum is a big old building or in basements, the audience may not have access to cellular data or wifi. But this approach sacrificed flexibility and maintainability, that if a museum wants to update its exhibits on display, it may need to update its application, and re-upload to Google Play Store or App Store after the updated application passes examination. This approach strongly violates the design goal that we should provide a highly maintainable workflow, and after balancing the advantages and shortcomings, we choose to use QR code to make AR MUSE Viewer App an online application. With the utilization of the QR code, our AR MUSE Viewer App can easily fetch models stored on the server, and users can save a large amount of time when they try to download for the first time.

As we mentioned above, the most fundamental function of this viewer app is the QR code scanner, so the open-source, multi-format 1D/2D barcode image processing library ZXing (ZebraCrossing) is the best choice. This library offers a stable and accurate QR code recognizing and tracking function that gives us a solid foundation and enables us to develop further (we will discuss this in the Viewer App Function part).

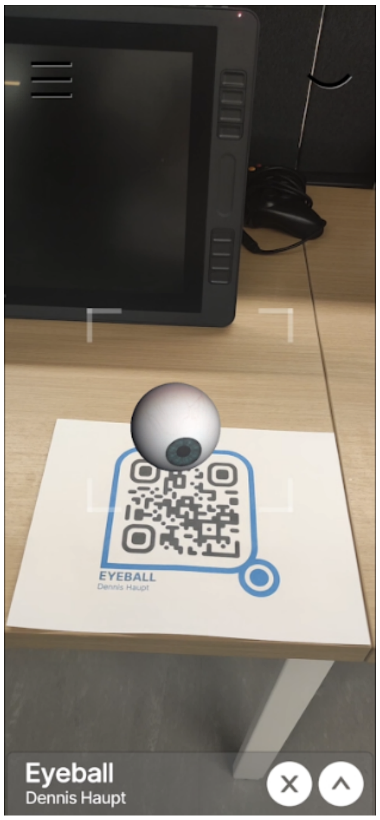


Figure 10: When Viewer App Scans a QR Code

3.3. Interaction and Experience Design

The core of designing a complex interactive experience is isolating and understanding the types of interactions. The AR MUSE system covers multiple types of interactions:

- How user interacts with space
- How user interacts with virtual exhibit
- How user interacts with the mobile Viewer App
- How user interacts with the AR MUSE Unity Toolkit

These types of interactions are drastically different, so it's challenging for the team to recognize and follow related design principles while balancing the complexity and customizability of the system.

3.3.1. Interacts with Space: QR Code

The first step to setting up an AR museum is organizing the space. We choose QR codes as references for virtual exhibits because they provide a reliable and consistent way to link digital content with the physical world. QR codes are easy to scan with mobile devices, and they can contain a significant amount of information in a small space. This makes them a practical and convenient way to trigger AR experiences.

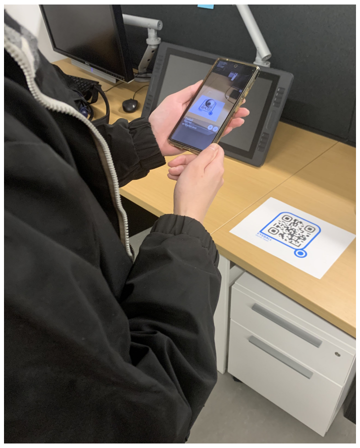


Figure 11: When Viewer App Scans a QR Code

Additionally, using a QR code significantly streamlined the workflow for setting up the exhibition. A QR code is highly customizable and could contain context-specific information. This advantage is the key to the design of the AR MUSE Unity toolkit. The QR code could serve as a "tag" that links to the package uploaded to servers. Unlike when using image tracking, the user has to pre-store the targeted image information. With a QR code, the user just needs to type in the "tag" with any QR code generator.

Furthermore, using QR codes as a reference point for AR can also enable tracking and localization. By scanning the QR code, the AR app can identify the user's location and orientation to the QR code, which can help align the digital content with the physical environment more accurately. This can enhance the realism and immersion of the AR experience.

3.3.2. Interacts with Virtual Exhibit: Viewer App

The Viewer App helps the user interact with digital exhibits with the help of a camera and QR code tracking. Beyond the basic function, the Viewer App also takes on the responsibility of enhancing immersion, providing additional information, and enabling remote access.

The Viewer App design provides a more interactive and immersive experience by applying animations to the exhibits and offering spatial interactions. The user could rotate the phone or the QR code to see the exhibits from different perspectives.

One of the most important features of the Viewer App is providing additional information about exhibits by overlaying text descriptions onto the user's camera view. The app could overlay the information input from the AR MUSE Unity toolkit into the camera view through an interactive menu. The user can click to see broad or detailed information while keeping the exhibits visible for observations and interactions.

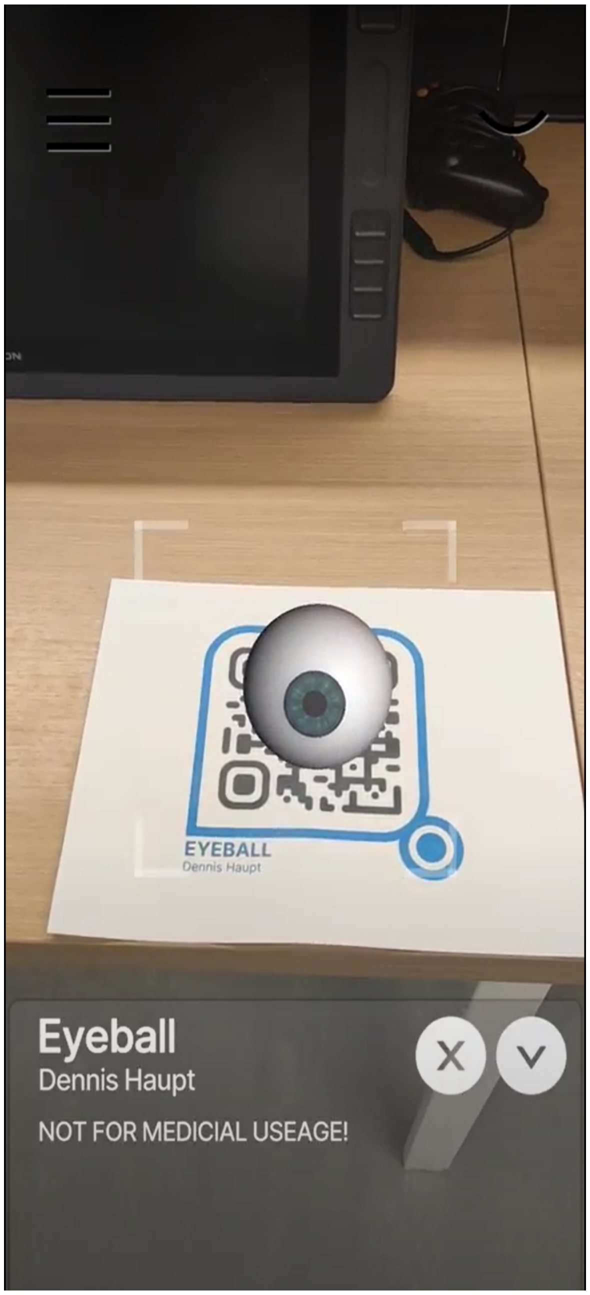
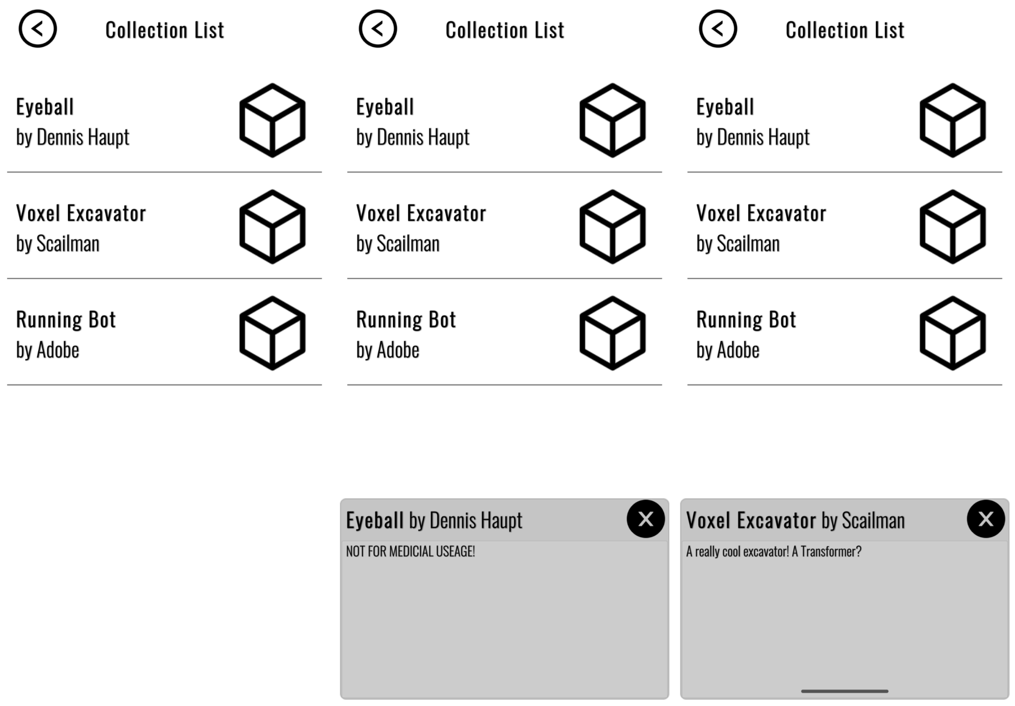


Figure 12: Text Description Overlay

The Viewer app can enable remote access to exhibits, allowing users to view them from anywhere through the museum code. This can be useful for users who are unable to visit the museum in person or for educational purposes.



NOT FOR MEDICIAL USEAGE!

A really cool excavator! A Transformer?

Figure 13: Collection List Page for Remote Access

3.3.3. Interacts with the Mobile Viewer App: Viewer App UI Design

The UI design of the Viewer App follows the Human-Computer Interaction principles. There are some notable keys: Usability, functions, accessibility, and performance.

The Viewer App's user profile has a broad audience. As a result, the App should be easy and intuitive to use and navigate, even for users who are unfamiliar with AR technology. This requires simplifying and streamlining some aspects of the app, such as the navigation menu and the process for accessing exhibits. Additionally, the App should also provide hints for the user, which are important in first experience.

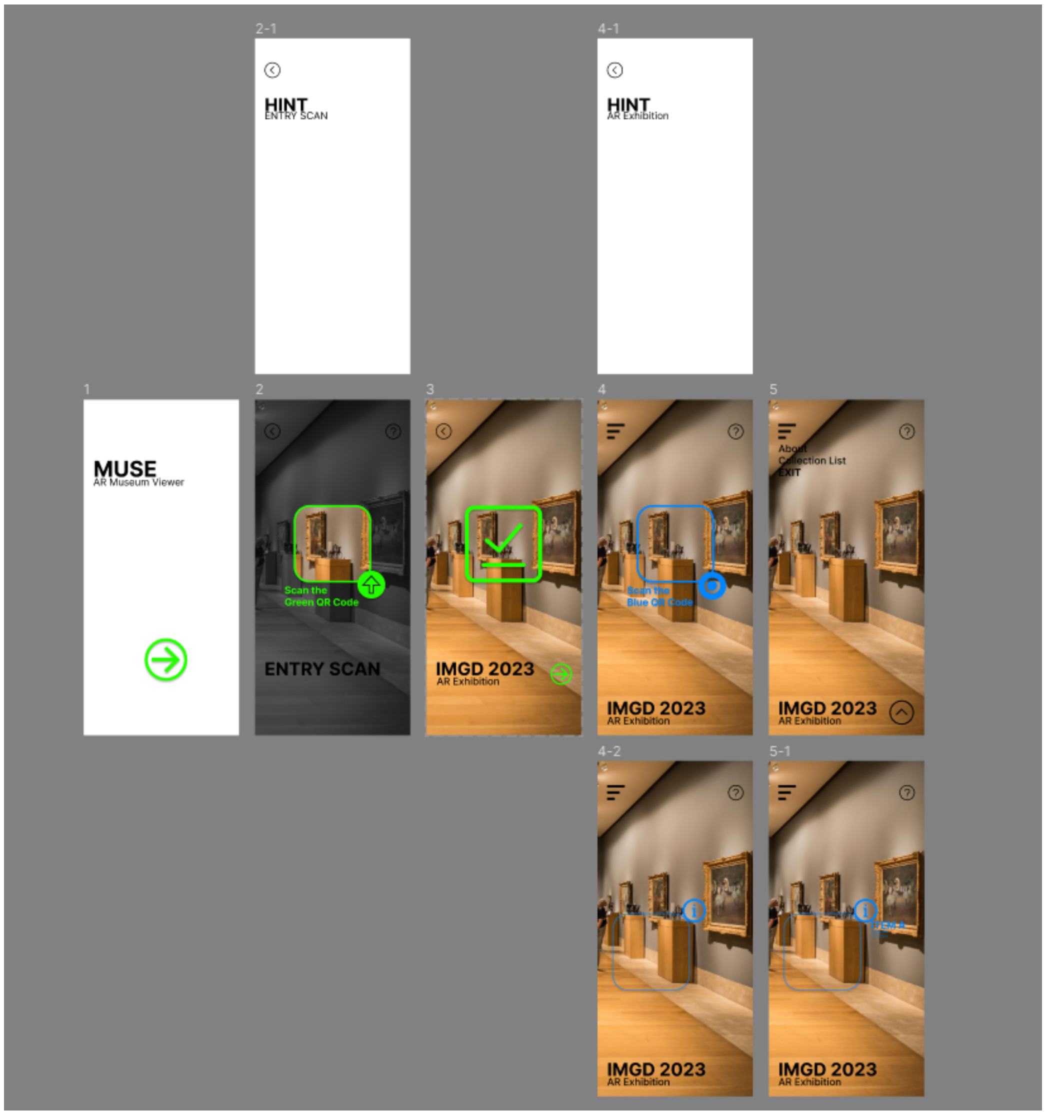


Figure 14: Viewer App Early Design

The main functions for the Viewer App are downloading assets from the server, presenting the digital exhibits, and providing additional information. The downloading functions are all hidden in the background in order to streamline the user experience. The presenting mode keeps the camera on for all time to create an immersive and interactive experience. The information menu is designed as a folded area. This design not only presents full information in a limited screen area, but also creates interactions with the user.

The Viewer App should be accessible to a wide range of users, including those with disabilities. The team went for a minimalist high contrast style with only black&white visual design. This helps the users with color vision deficiency use our system with ease. Additionally, haptic feedback is used in the QR code scan process to notify the user that the scan is successful. The lightweight interactions from the View App helps the user use the system confidently.
In order to ensure the App performance on different mobile platforms, we minimized the Viewer App function and UI animations. This design decision is consistent with the app's visual design while offering clarity and simplicity.

3.3.4. Interacts with the Unity Toolkit: Unity Toolkit Design

After we abandoned the Curator App, the user would have to access Unity Editor to use the curator functions. Although the Unity editor creates a usability and technical barrier for general users, the team tries to guide non-professional users by designing the Toolkit UI carefully.
For clarity and simplicity, the Toolkit is integrated with the Unity Editor on the top menu. The functions are streamlined into two major steps and all the elements are organized in a logical way. The buttons and menus are presented with clear and descriptive labels, with a clear hierarchy of screens and sections.

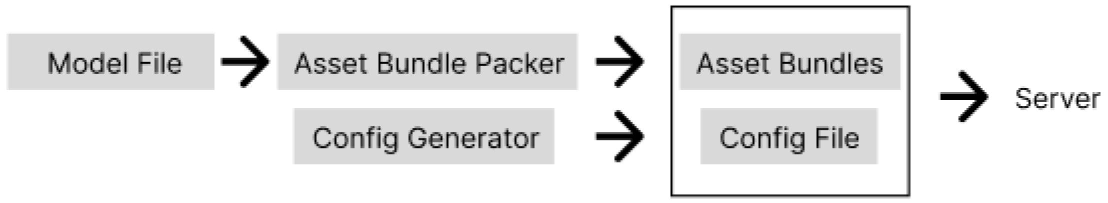
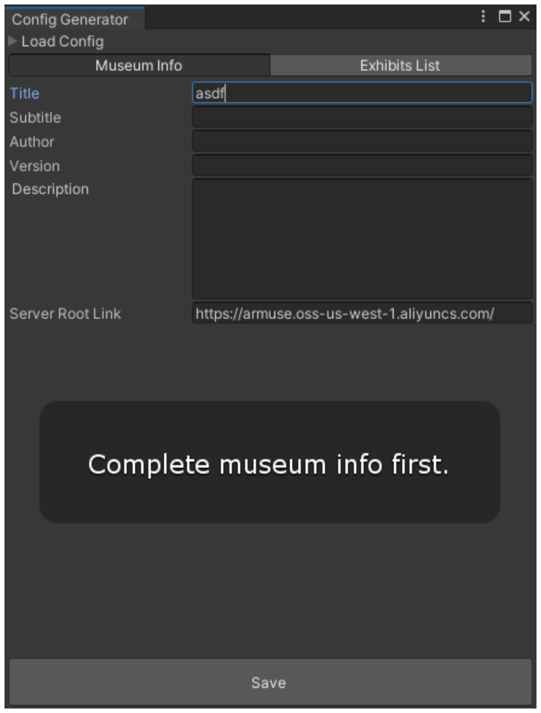


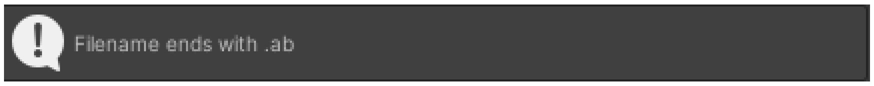
Figure 15:Unity Toolkit Workflow

The UI design is also consistent across different screens and sections and follows the Unity Editor design. This can help users to navigate the toolkit more easily and reduce confusion.
The Toolkit also provides feedback to users when they interact with detailed error messages. A step-by-step readme file is also included for non-professional users.



Title asdf
Subtitle
Author
Version
Description
Server Root Link https://armuse.oss-us-west-1.aliyuncs.com/

Complete museum info first.



Filename ends with .ab

Figure 16: Error messages

The balance between complexity and customizability is also a key point to consider. The team decided to take advantage of the powerful 3D model edit and animation function in the Unity Editor while keeping our AR MUSE Unity Toolkit simple and direct. As a result, the user can use all the complex functions in the editor and save them to a prefab. The Toolkit then takes the prefab to the server and sends it to the Viewer App.

3.4. AR MUSE Implementation

Before stepping into the implementation section, we need to have a brief introduction of this solution. AR MUSE contains two parts: an AR MUSE Unity Toolkit to create and manage exhibitions, and a Viewer App for exhibition audiences of the museum to inspect these models.

This solution also needs two external platforms: an Object Storage Server (OSS) for storing all the packed models and config files, which is specifically designed for Viewer App to read and download remotely, and a QR code generator, that user can generate QR codes with any pattern, and then link QR codes to the models stored on server manually.

3.4.1. Unity Toolkit Implementation

AR MUSE Unity toolkit serves as a tool to help creators manage and create exhibition areas. One of the functions includes the QR code coordinate reference tool we mentioned earlier. This helps with the Creator to have a preview of how their work will look like related to the QR code in real life.
The other main function is to pack all the exhibit Bundles, create and manage the config file of the exhibition area.

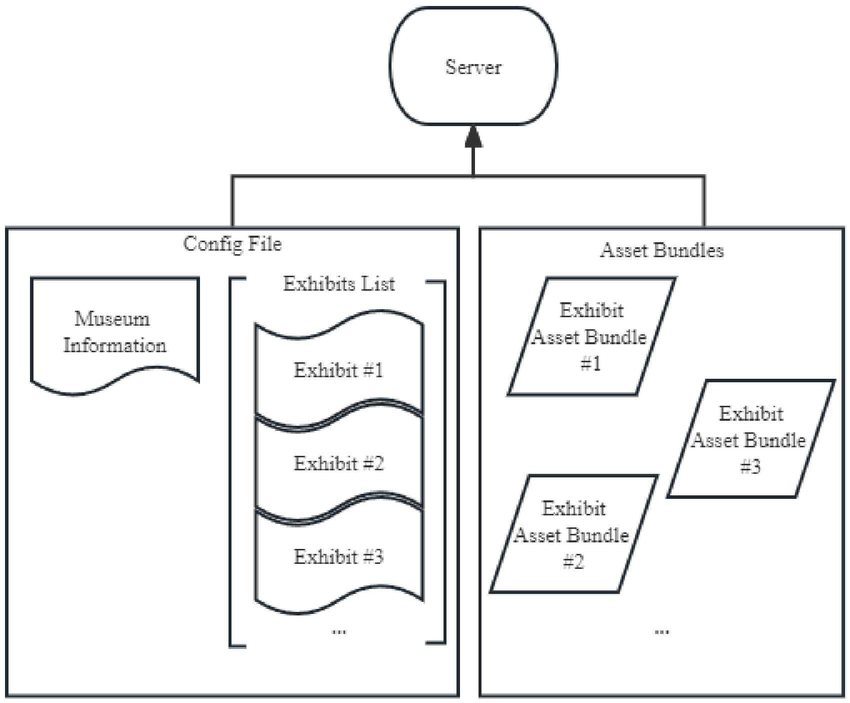
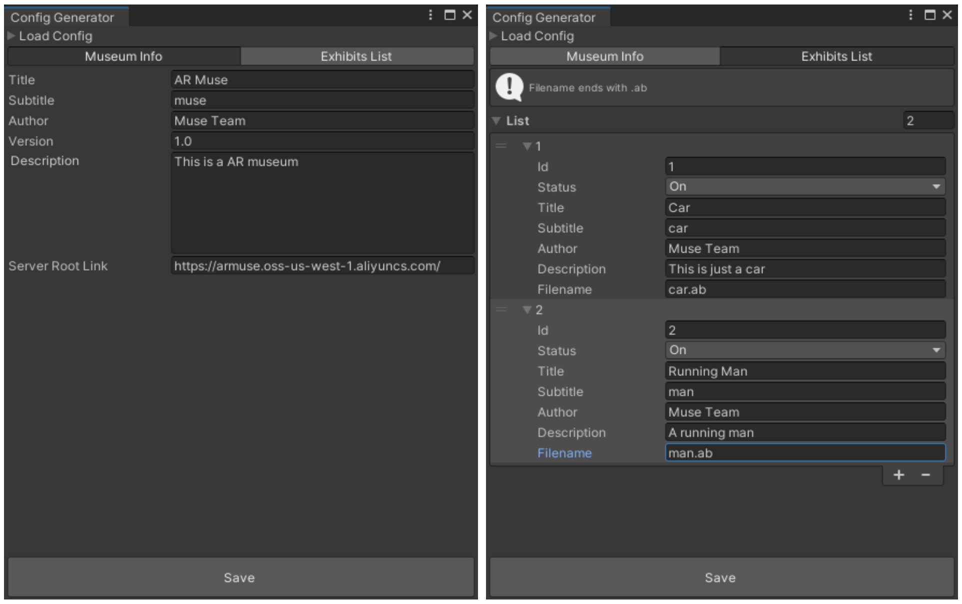


Figure 17: Upload Structure

The user will need to have a config file first. This file includes the information of the Museum and the information of each individual exhibit. The config generation tool in Toolkit has a simple and clear UI. The user only needs to fill in the corresponding information and click the SAVE button, and the config file will be generated.



Title AR Muse
Subtitle muse
Author Muse Team
Version 1.0
Description This is a AR museum
Server Root Link https://armuse.oss-us-west-1.aliyuncs.com/

Filename ends with .ab
List 2
1
Id 1
Status On
Title Car
Subtitle car
Author Muse Team
Description This is just a car
Filename car.ab
2
Id 2
Status On
Title Running Man
Subtitle man
Author Muse Team
Description A running man
Filename man.ab

Figure 18: Config file

After all the textual information has been entered into the config file, the user also needs to package all the actual model files. A package tool is also included in our toolkit, users only need to drag all model files into the window then press the BUILD button, the toolkit will complete the packaging at one time.

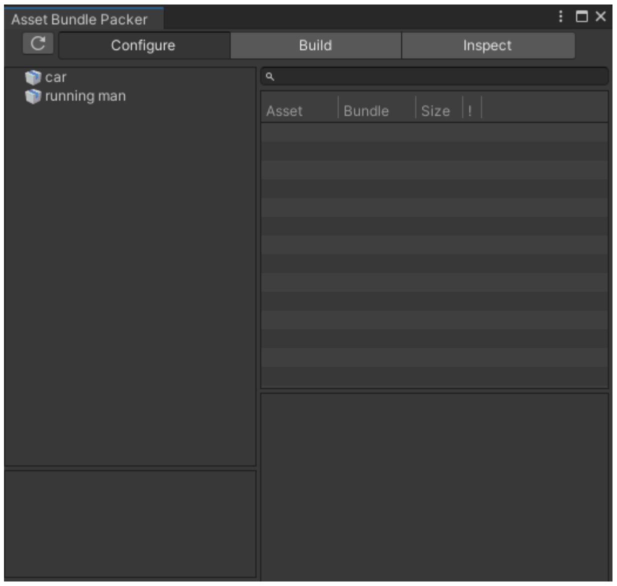


Figure 19: AssetBundle Packer

3.4.2. Viewer App Implementation

The Viewer App is also developed in Unity Engine, and only supports Android platform. The Viewer App starts with scanning the Entry code, the scan process is shown in figure 20.

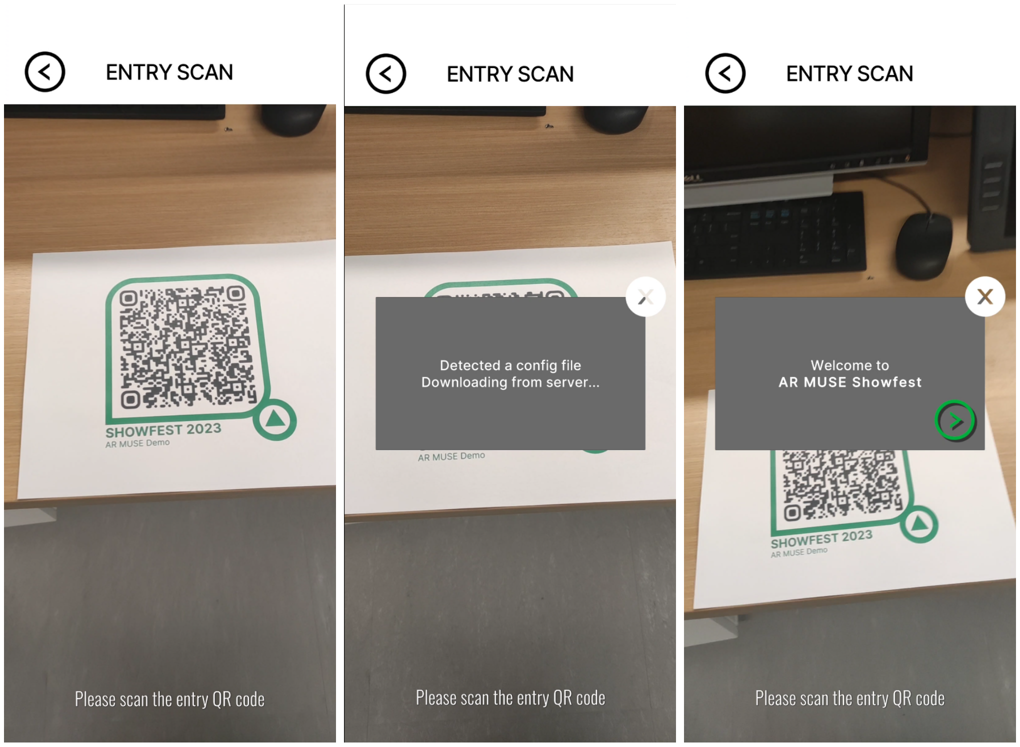


Figure 20: Entry Code Scan Progress

When the Viewer App scans a code, it will automatically read its content, and if the content passes detection that it is a valid link directed to a xml file, it will try to download that file. The downloaded file will be analyzed next to get all the elements, and then Viewer App will start a data persistence process to make data available when Viewer App enters other sessions. All the data will be stored in the PlayerPrefs class, which is a class that stores Player preferences between game sessions[18]. The data persisted will be used for individual QR code to get a full link directed to each model.

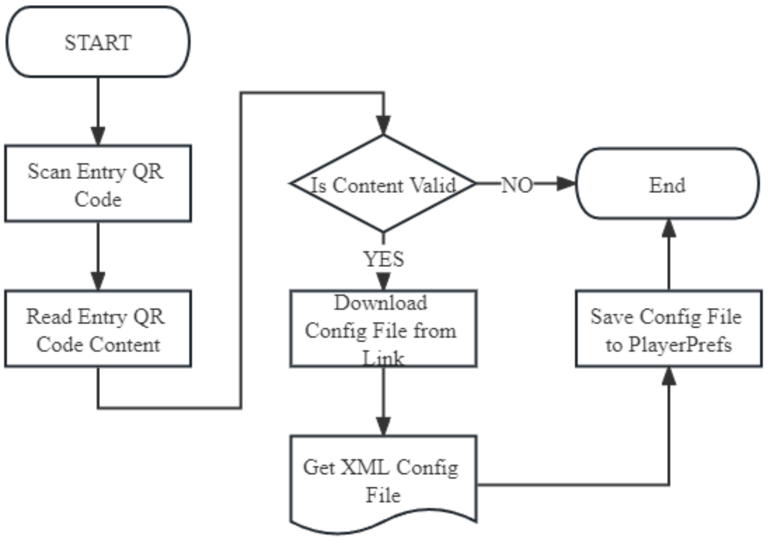

START
Scan Entry QR Code
Read Entry QR Code Content
Is Content Valid
NO
End
YES
Download Config File from Link
Save Config File to PlayerPrefs
Get XML Config File

Figure 21: Entry Code Scan

After successfully stored the data, user can entry the main session of the Viewer App, which looks like this:

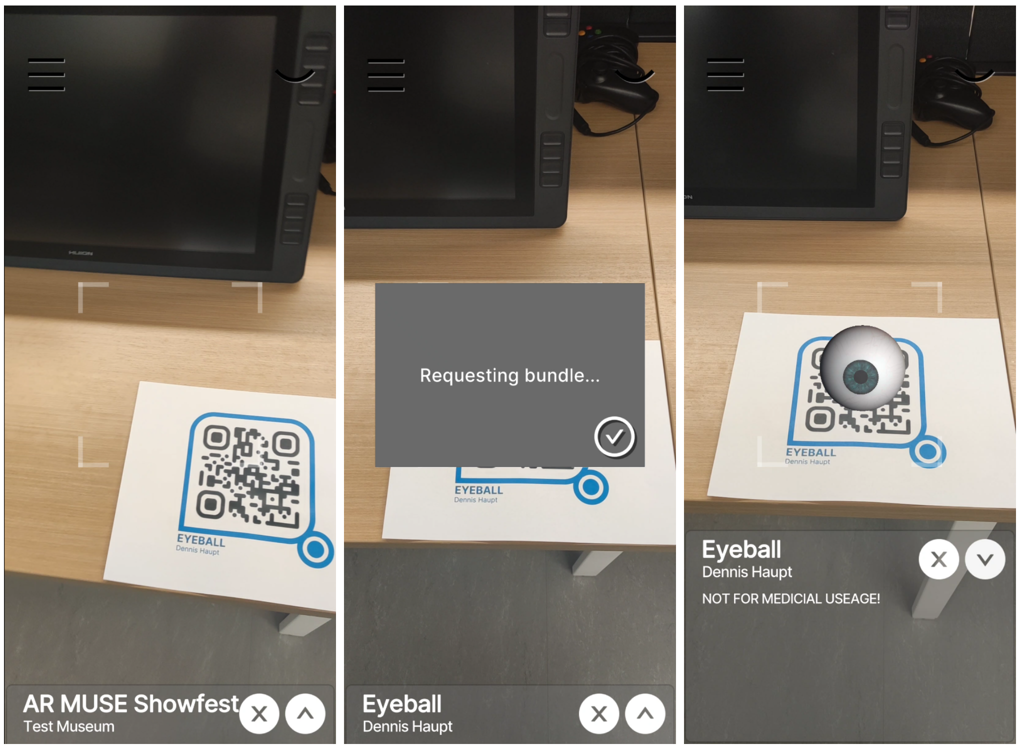


Figure 22: Main Session

The content of each individual QR code is just the filename of these models saved on the server, for example the eyeball model shown before, the QR code's content is “eyeball.ab”. And we will talk about how a filename connects to the models uploaded since we do not link them directly in a config file. The model’s link is consists of four parts:

- Server-root-link: The Viewer App will get the server’s root link from persisted data, which was saved in the PlayerPrefs class before, and transferred to the main session.
- Asset-bundle: This is the default string that will automatically append after the server root link string.
- Platform: Because asset bundles are platform-specific, which means it is not allowed to load a not compatible asset bundle. So, the Viewer App will detect the audience's cell phone platform, if it is Android platform, it will append “android”, or if it is IOS platform, “ios” instead.
- Model-file-name: This is where the Viewer App recognizes the QR code linked model. There is a point that two files with the same filename will never exist in a directory. In OSS, if a creator unintentionally uploads multiple files with the same filename, the system will automatically rename them by appending additional string like “(1)”. This operation makes each model have its unique filename, and naturally times out when an audience scans a non-existent QR code.

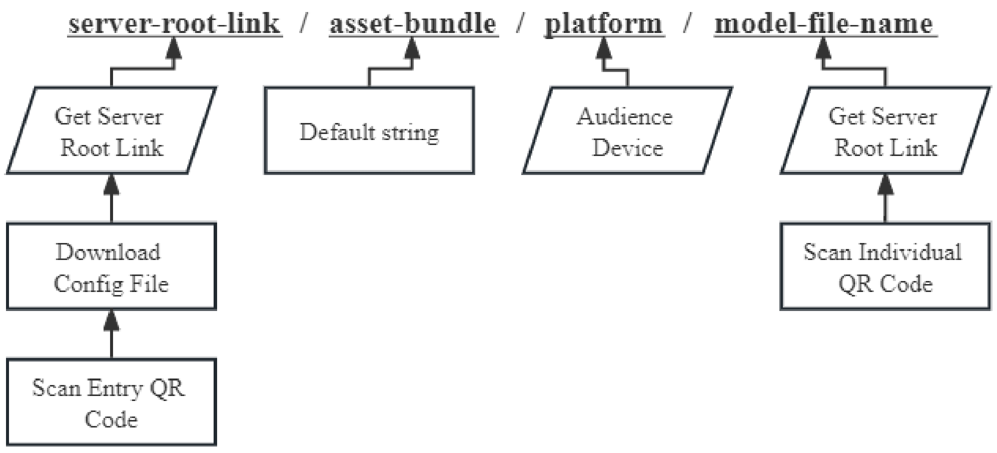


Figure 23: Link String Append

4. Evaluation

4.1. Early Development Test

During the initial development stage, we brought our first version of the Viewer App to Alphafest for testing and feedback. Based on our observations, interviews, and collected questionnaires, we found that the overall design of the app was successful, but there were still many details that needed improvement. For instance, first-time users had difficulty navigating through the app and determining whether the model had been loaded. To address these issues, we streamlined the components of the Viewer App and added more guidance, haptic feedback, and hints to the app.

4.2. Evaluation Design

After confirming the target group of users and analyzing the needs of them. We designed an usability test to confirm that our system has indeed lowered the threshold for creating AR exhibition. And people who are interested in creating this AR project really feel that our system/tool has made the work easier. The whole test has 4 phases:

1) Pre-Test Survey
2) Toolkit Test
3) Viewer App Test
4) Post-Test Survey

4.2.1. Pre-Test Survey

This part of the Survey is mainly a background investigation on the AR-related background and Unity background of the testers. It helps teams better characterize testers and evaluate their performance individually.

4.2.2. Toolkit Test

This part of the users was given a task, which was to use our toolkit to create an AR Exhibition. The testees were provided with two .obj model files. They could pick one, or both, to create an AR exhibition that included this work or both works. We provided a guidance document [appendix] to let the test personnel learn how to use the system. We observed the whole process to see if the testers followed our expected plan or encountered any difficulties we didn't expect. The whole process, from the time when the tester started using the toolkit to when the project was packaged, was timed.

4.2.3. Viewer App Test

After the tester finished creating their own AR Exhibition, they were told that they could use our viewer app to observe the work they had just completed. Because everyone's model editing was slightly different, even though they were all provided with the same model file, the final results varied. This stage was mainly to test the usability of the Viewer app and whether it met user expectations.

4.2.4. Post-Test Survey

4.2.4.1. Quantitative questions

We used the System Usability Scale[19] to do quantitative evaluations. The System Usability Scale (SUS) is a standardized questionnaire that is widely used to evaluate the usability of a wide range of products and services, including software, websites, mobile apps, and other interactive systems. It was developed by John Brooke in 1986 and has become one of the most popular and widely used tools for measuring the usability of interactive systems.

The SUS consists of a 10-item questionnaire that measures the user's perceptions of the usability of the system. The questionnaire consists of five positive statements and five negative statements, and users are asked to rate their level of agreement with each statement on a five-point Likert scale, ranging from "strongly agree" to "strongly disagree." The scores are then aggregated and converted into a usability score ranging from 0 to 100.

A SUS score of 68 is considered average, and scores above 68 are considered above average. Scores below 68 indicate that there may be usability issues with the system that need to be addressed.

Additionally, the team can analyze the scores for each individual item on the questionnaire. This can help identify specific areas of the system that may be causing usability issues. We can also compare the SUS scores with pre-questions to get a better image of our tester and how our system performs in the hands of users with different Unity and computer science backgrounds.

4.2.4.2. Qualitative Questions

This part of the evaluation is in the form of linear scale and short answers. This part of the evaluation helps us get more opinions from the testers regarding the highlights of our system.

4.3. Evaluation Result

4.3.1. Quantitative Questions Result (System Usability Scale)

Table 2: Background Survey and System Usability Scale Survey Result

	Test Time (s)	Unity Skill Self Assessment (out of 5)	I think that I would like to use this system frequently.	I found the system unnecessarily complex.	I thought the system was easy to use.	I thought that I would need the support of a technical person to be able to use this system.	I found the various functions in this system were well integrated.	I thought there was too much inconsistency in this system.	I would imagine that most people would learn to use this system very quickly.	I found the system very cumbersome to use.	I felt very confident using the system.	I needed to learn a lot of things before I could get going with this system.	System Usability Scale Score (out of 100)	
	5:37	3	3	3	4	2	3	1	4	3	2	2	62.5	
	9:02	2	3	1	5	3	4	1	4	2	4	3	75	
	8:09	3	5	2	4	3	4	2	5	1	3	2	77.5	
	9:03	3	3	3	3	4	2	2	3	2	2	3	47.5	
	7:20	4	2	1	5	2	5	1	4	2	4	1	82.5	
	9:00	4	2	1	5	4	4	1	4	1	4	2	75	
	9:00	2	4	2	4	3	4	2	4	2	5	2	75	
	10:53	2	2	1	4	4	4	2	5	3	5	4	65	
	4:35	3	3	1	4	3	4	2	4	1	2	1	72.5	
	4:46	3	3	2	4	2	4	1	4	1	5	2	80	
														STD
Mean	7:44		3	1.7	4.2	3	3.8	1.5	4.1	1.8	3.6	2.2	71.25	10.35816908
Median			3	1.5	4	3	4	1.5	4	2	4	2		
Mode			3	1	4	3	4	1	4	2	2	2		
STD			0.9428090416	0.8232726023	0.632455532	0.8164965809	0.7888106377	0.5270462767	0.5676462122	0.7888106377	1.264911064	0.9189365835		

Typically, a SUS score of 68 is considered average. The mean SUS score for our system across 10 test results is 71.25 and the standard deviation is 10.36, 14.5% of the mean value. However, a score of 47.5 is considered an outlier. Overall, the SUS score shows our system usability is above average.

Looking into each question and considering their positive and negative wording, there are three questions show below average or scattered results: “I think that I would like to use this system frequently”, “I thought that I would need the support of a technical person to be able to use this system” and “I felt very confident using the system”. These results could be caused by different backgrounds of the testers. It will be discussed in the discussion sections.

4.3.2. Qualitative Questions Result

See Appendix for interview questions and response record.

The following are highlights for the open questions:

1. The system is proven easy to use for people with different technical backgrounds
    - “I overall had a very positive user experience even as someone who is not familiar with Unity. I found the instructions a crucial and effective component of the process which helped me design something and bring it to life. I thought the app was very user friendly!”
    - “It's very easy to use and I think with this toolkit I can create an AR project very quickly.”
    - “Nothing but praise, well done!”
2. Users rated the UI design & integrated workflow favorably.
    - “It's very easy to use and the UI is very clean and good to understand.”
    - “Having never developed AR technology, I was a little worried about complexity, but the system made the process very intuitive through unity and the custom editors.”
    - “I like the UI~”
    - “How easy it was to transfer the data from unity into a QR code was very impressive.”
3. Subjects imagined diverse uses for AR MUSE
    - “I think it could be really helpful in teaching environments for things like understanding molecules and other models.”
    - “Sharing a 3D modeling portfolio.”
    - “Quick visualization for a combined 3D scene.”

- “I'd use it to view my 3D models from Zbrush/Maya if I got them into Unity.”
- “I would use this system if I was developing a piece of art (game assets or actual art) to allow for those to seemingly come to life in AR could be valuable to provide users with a unique experience.”

4.4. Discussion

Based on the evaluation results, we believe we have successfully achieved our design goal of reducing the barriers to entry for creating AR projects. Most people find our system easy to use, especially one of the testers who has experience in AR development and found the system very helpful. We acknowledge that some testers may have underestimated our system to some extent because they lacked experience in AR development, making it difficult for them to compare our system to others in terms of making the development process easier. Nevertheless, even those who had no experience in AR development were able to complete the tasks we assigned them with a written guide in a very short time, with an average time of under 10 minutes, which is great news.

Table 3: Test Time, Unity Skill Self Assessment and System Usability Scale Score

Test Time (s)	Unity Skill Self Assessment (out of 5)	System Usability Scale Score (out of 100)
5:37	3	62.5
9:02	2	75
8:09	3	77.5
9:03	3	47.5
7:20	4	82.5
9:00	4	75
9:00	2	75
10:53	2	65
4:35	3	72.5
4:46	3	80

It is interesting to note that the responses for the questions "I think that I would like to use this system frequently," "I thought that I would need the support of a technical person to be able to use this system," and "I felt very confident using the system" have a higher standard deviation. After considering the positive and negative wording of the questions, we observed that although there is some very positive feedback in each question, the mean scores were lower than expected. In another word, although some tests have very positive feedback, testers in general have very different opinions on these questions. We believe that the variation in scores could be due to the subjective nature of the questions and the differing technical backgrounds of the users. We also tried to see if the tester's familiarity with Unity would influence their ability to learn and use our system. Because this is a toolkit based on Unity. However, we found that relying on the testers' self-assessment of their Unity skill level was not very accurate. Some testers who completed the task much faster than others did not consider themselves to be very skilled in Unity. This indicates that simply asking about Unity skill level may be too broad and not provide an accurate assessment of the testers' abilities. Therefore, trying to obtain an accurate assessment of the testers' Unity level may not be necessary in this case.

Table 4: Notable System Usability Scale Questions Results

	I think that I would like to use this system frequently.	I thought that I would need the support of a technical person to be able to use this system.	I felt very confident using the system.
Mean	3	3	3.6
Median	3	3	4
Mode	3	3	2
STD	0.9428090416	0.8164965809	1.264911064

5. Post-mortem

5.1. Achievements and Lessons Learned

We also learned from the abandoned Curator App. As mentioned in the previous article, we abandoned this app. The major reason why we canceled it it's because of an insurmountable technical problem: Unity can’t save .ab files in the runtime. We didn’t realize it at first because we managed to save a file and Unity also generates a .ab file successfully at runtime. But after we upload the file to the server it just keeps failing to read. For a very long time we thought it was a server issue and spent a lot of time trying to fix this "server problem". Eventually we found out it wasn't a server issue. The .ab file is actually fake. Unity can generate a .ab file at run time, but there is nothing valid in it. Unity can store .txt, .png or even .obj files at runtime but it can't pack .ab files. This is the underlying design of Unity. We can't do anything to fix it unless we not keep using Unity.

One of the reasons to completely abandon this App is the lack of a valid reason to continue using it. During the early stages, we informally tested the app and realized that we had underestimated our users' capabilities. We mistakenly assumed that artists, especially traditional artists, may lack technical skills, including model editing. However, through informal interviews with several users, we learned that modern or electronic artists possess these technical skills, and some even master them. Therefore, simplifying the model editing function is unnecessary and may even limit their creativity, leaving some users feeling frustrated. Our original design was to provide advanced editing through a toolkit, but we overlooked the fact that the original toolkit was too complicated for them. We failed to consider what artists truly need and misanalysed their requirements. In reality, artists need a simplified solution for binding art pieces with QR codes.

Thus they really care about the relationship between the QR code and the model, people like the reference coordinate for the QR code so we kept and improved this idea to the toolkit. We know that QR codes are directional, and we think that's some kind of common sense, but people don't really think so. Thanks to the early test, we figured this question out and improved this by adding an arrow in the coordinate.

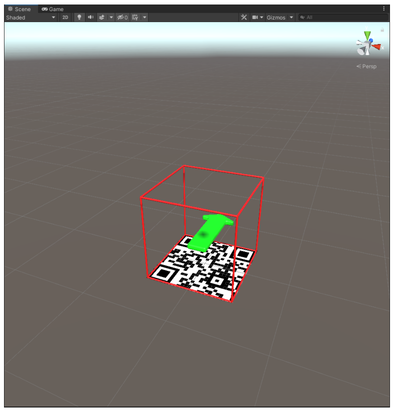


Figure 24: Coordinate Reference in Toolkit

This system also has a limitation that cannot be bypassed. The entire system theoretically requires users to have access to the network. The requirements for the creator will be higher. The creator not only needs to have network access, but also needs to have access to a server to store the works. (Note that the creators mentioned here can be individuals or organizations.) This limitation comes from our original idea, hoping to get rid of storing a large number of files locally on the mobile phone.

For most current common AR App, users need to download a large file package in one place including all the data. In this case, users usually choose to download in a Wi-Fi environment, so that they do not have to worry that their mobile data plan will run out. Our viewer App is designed to be as lightweight as possible. Users download data from the server when they need them. This may cause some data problems, a mobile phone needs to download from the server every time it scans a QR code. However, in theory, the data contained in the QR code is an address that points to where the model is stored, it is also feasible to use a local Wi-Fi network. The internet speed will also affect the viewing experience of the user. Although usually a file with a size of tens of MB can be displayed within 1 second. But in the worst situation the user's own mobile network signal is not good or the network speed is very slow. The exhibitor is suggested to provide Wi-Fi to ensure that the network is the best.

Another limitation is the scanning limit of QR codes. To see the work link to the QR code, user need to first scan the QR code. After that, the user only needs to keep the QR code on the screen after scanning, the viewer app can detect where the QR code is and will track the QR code to correct the model position. Even if the QR code is lost, the phone will use the gyroscope to identify the space, but it won't be very accurate. However, in order to accurately read the information in the QR code, the user must face the QR code and keep an appropriate distance from the mobile phone when scanning. If there are too many viewers for the same work, they may have to queue up to scan the QR code.

5.2. Future Work

After considering the test results and observing the test process. It’s important to make the system more accessible for users with little Unity or AR development background.

- A more detailed user manual for both Viewer App and AR MUSE Unity Toolkit
- Include more built-in hints in the Viewer App and AR MUSE Unity Toolkit

For a shippable product, we probably need a more detailed, step by step tutorial for how to use this system. People also think the viewer app can be prettier. More animation effects in the Viewer app is another stuff people mentioned in the test. The UI can still be improved.

Finally, this will be an open-source project. We have published our work on Github: https://github.com/dmql98/AR-Muse-System.

5.3. Conclusion

When an artist wants to hold a personal exhibition, they usually need to cooperate with local museums and pay venue fees, which is not economical for starter artists. However, our project, AR MUSE solution, enables artists or museum curators to create an AR museum in a relatively low budget, and more importantly, limited knowledge will no longer be the barrier. With Unity Engine, they can modify their amazing models in the simulated environment, and use our team’s toolkit to simplify the process of deploying these models to a remote server, which will be linked by QR codes. Our team also developed a mobile app for inspecting these works by scanning QR codes.

This solution achieves its design goal according to the test we conducted, that most of the participants have limited knowledge of processing with models, and experience with AR environment. We received lots of positive feedback from several perspectives, and most significantly, the QR code mechanic, which is the foundation of our solution.

Bibliography

[1]Kan, Tai-Wei, et al. “Applying QR Code in Augmented Reality Applications.” Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry, 2009, https://doi.org/10.1145/1670252.1670305.

[2]Tiwari, Sumit. “An Introduction to QR Code Technology.” *2016 International Conference on Information Technology (ICIT)*, 2016, https://doi.org/10.1109/icit.2016.021.

[3]Laricchia, Federica. “U.S.: Tablets Average Price 2013-2027.” *Statista*, 28 Feb. 2023, https://www.statista.com/statistics/619505/tablets-average-price-in-the-us/.

[4]“Pokémon GO.” *Pokémon Go*, https://pokemongolive.com/?hl=en.

[5]“Ingress Prime.” *Ingress Prime*, https://www.ingress.com/.

[6]“Burberry Invites Customers into the World of Olympia with Its Latest Augmented Reality Experience.” *Burberry Corporate Website*, https://www.burberryplc.com/en/news/brand/2021/burberry-invites-customers-into-the-world-of-olympia-with-its-la.html.

[7]Britton, Alan. “Skyview Stargazing App Review: Locate Stars and Planets with Your Smartphone.” *Space.com*, 22 Sept. 2022, https://www.space.com/skyview-review.

[8]“Froggipedia Review for Teachers.” *Common Sense Education*, https://www.commonsense.org/education/reviews/froggipedia.

[9]Morpholio. “Ar Sketchwalk.” *Collaboration Tools: AR SketchWalk - Morpholio Trace User Guide*, https://morpholioapps.com/userguide/trace/?How-to-use-ar-sketchwalk.

[10]“Ikea App Page.” *IKEA*, https://www.ikea.com/au/en/customer-service/mobile-apps/say-hej-to-ikea-place-pub1f8af050.

[11]“Vuforia Enterprise Augmented Reality (AR) Software.” *PTC*, 27 Mar. 2023, https://www.ptc.com/en/products/vuforia.

[12]Apple Inc. “Augmented Reality.” *Apple Developer*, https://developer.apple.com/augmented-reality/.

[13]“Ar Foundation: Ar Foundation: 5.0.5.” *AR Foundation | 5.0.5*, https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html.

[14]“Create Webar for Free - No-Code Online Platform for WebAR & AR.” *Web*, https://web-ar.studio/en/.

[15]“Reblink.” *Art Gallery of Ontario*, https://ago.ca/exhibitions/reblink.

[16]“AOL Augmented Art Gallery.” *YouTube*, YouTube, 1 June 2021, https://www.youtube.com/watch?v=ry1v2G1WoJo&t=1s.

[17]Lauren, Post author By. “.FBX VS .Obj - Vectorworks Export to Unity.” *Scenic Mentor*, 30 June 2021, http://scenicmentor.com/fbx-vs-obj/.

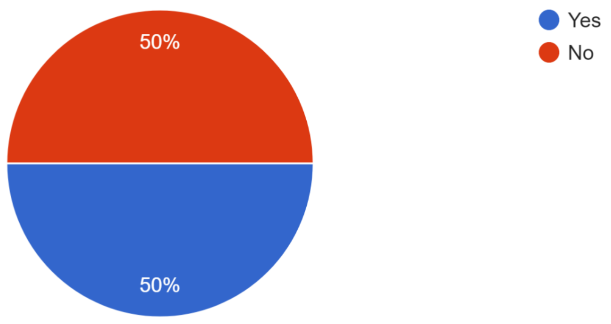
[18]Unity Technologies. “PlayerPrefs.” *Unity*, https://docs.unity3d.com/ScriptReference/PlayerPrefs.html.

[19]*Sus - a Quick and Dirty Usability Scale*. https://www.researchgate.net/profile/John-Brooke-6/publication/228593520_SUS_A_quick_and_dirty_usability_scale/links/5f24381392851cd302cbaf25/SUS-A-quick-and-dirty-usability-scale.pdf.

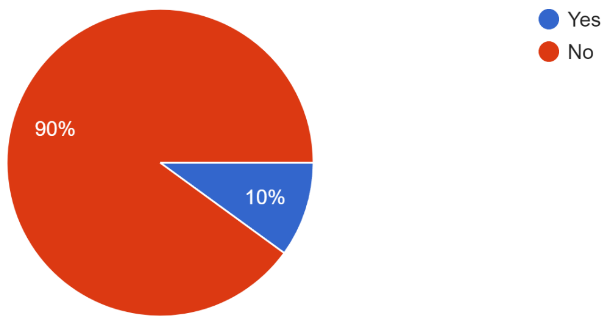
Appendix: Playtesting Questionnaire

1. Background Survey

I have experience with using mobile AR software
10 responses

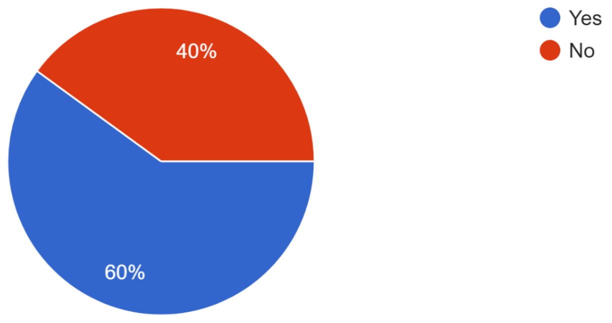


I have experience visiting one of AR Museum/Exhibit
10 responses



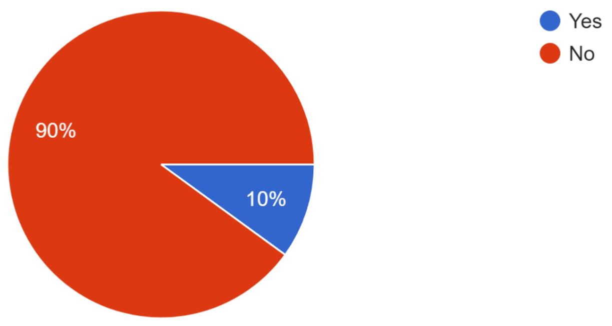
I am aware of AR Museum/Exhibit

10 responses



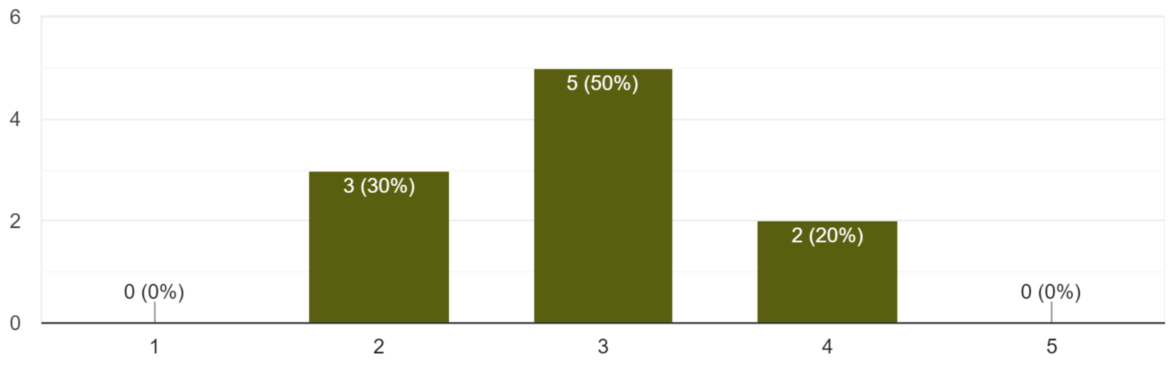
I have experience with developing mobile AR software

10 responses



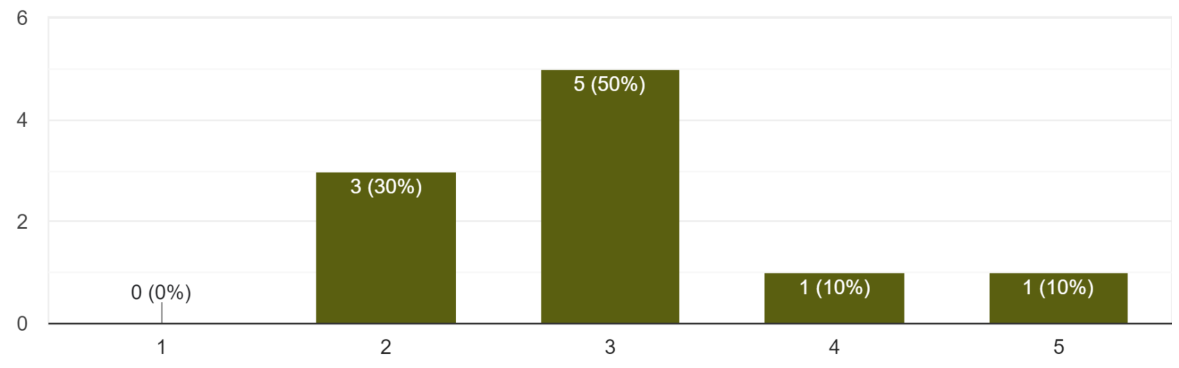
I am familiar with Unity

10 responses

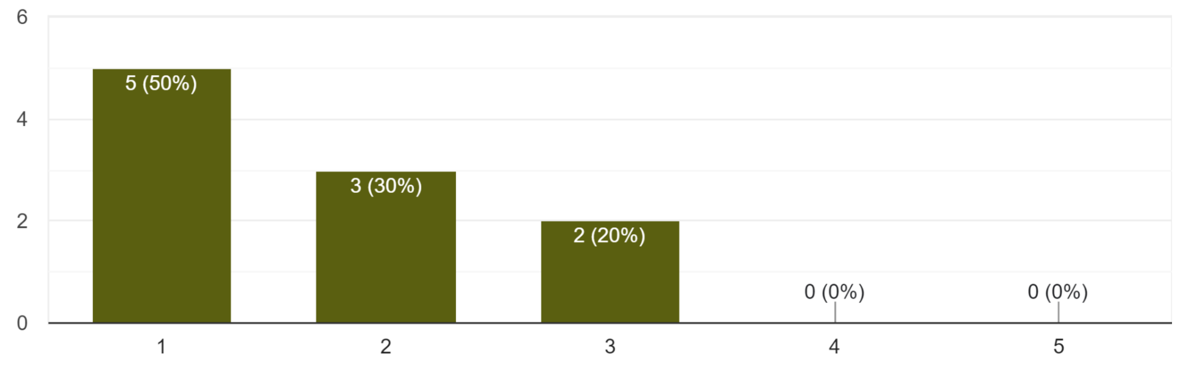


2. System Usability Scale

I think that I would like to use this system frequently.
10 responses

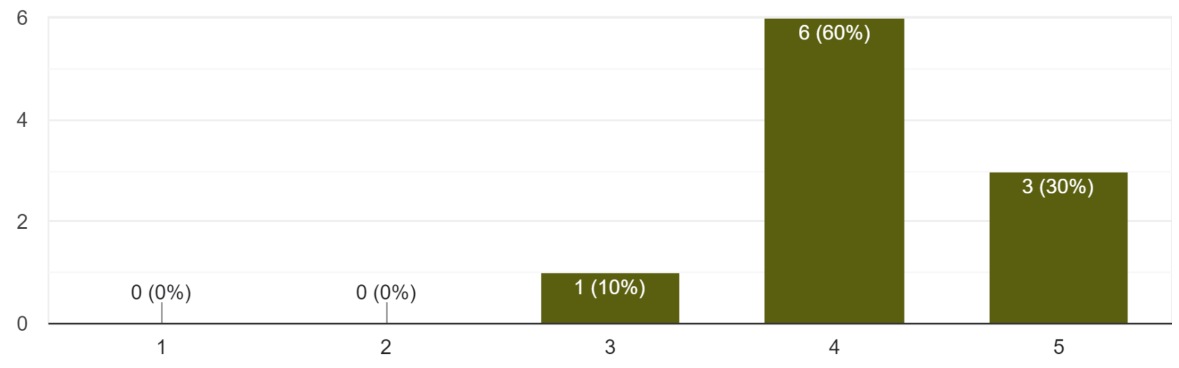


I found the system unnecessarily complex.
10 responses



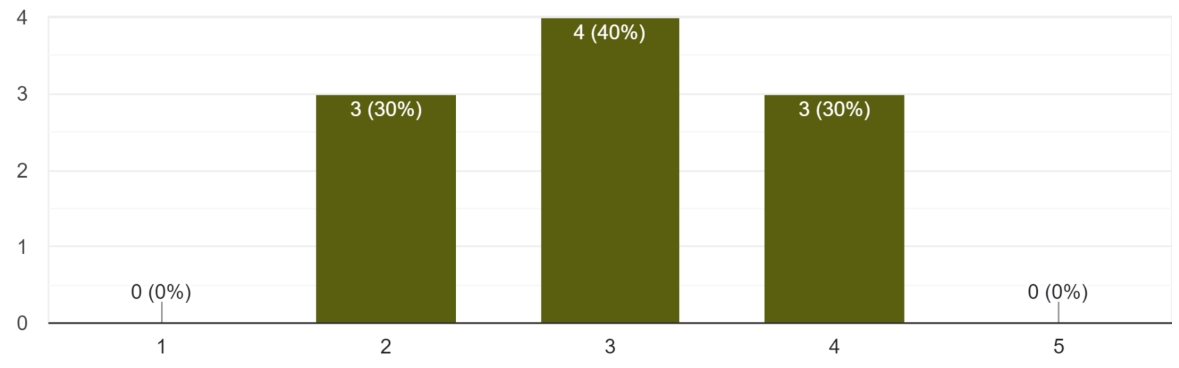
I thought the system was easy to use.

10 responses



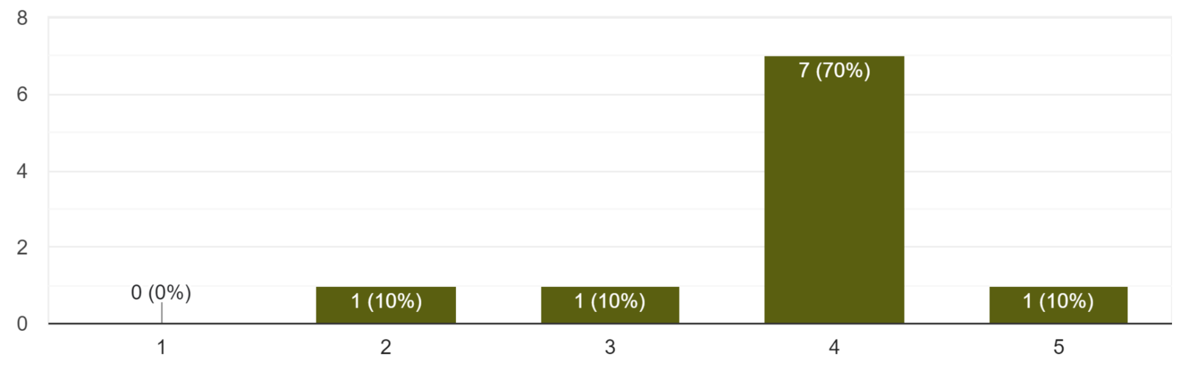
I thought that I would need the support of a technical person to be able to use this system.

10 responses



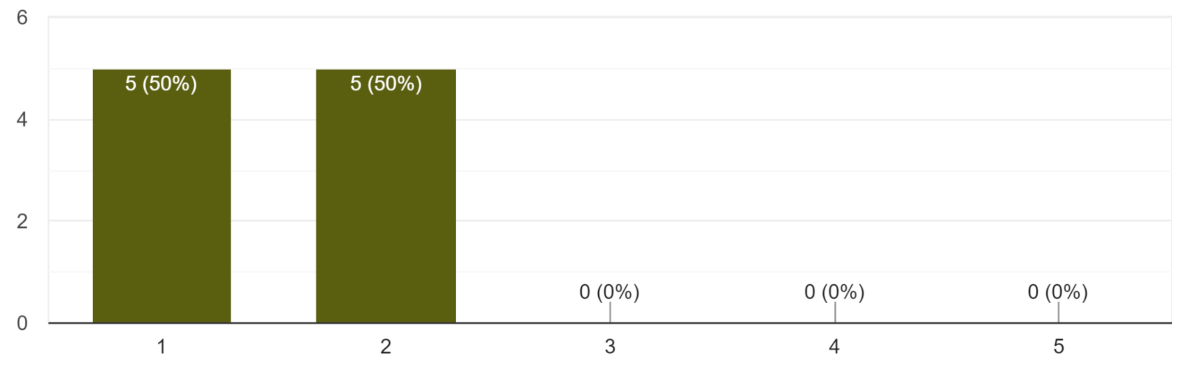
I found the various functions in this system were well integrated.

10 responses



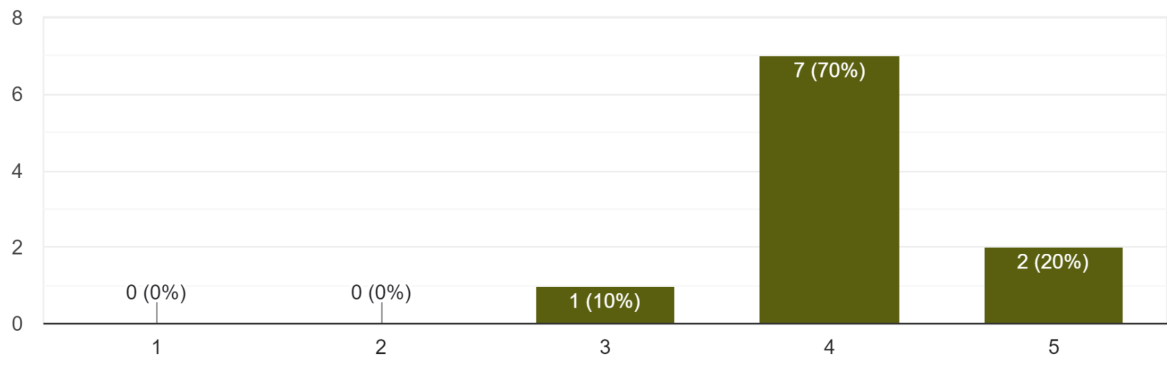
I thought there was too much inconsistency in this system.

10 responses



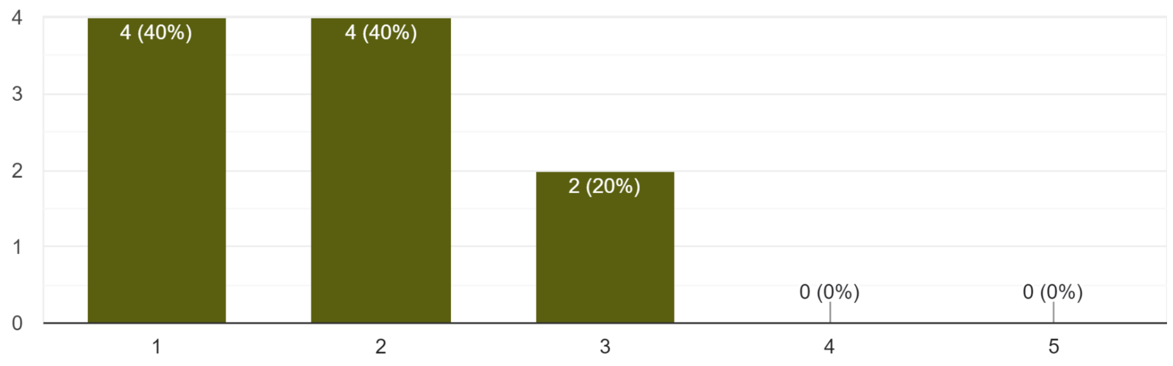
I would imagine that most people would learn to use this system very quickly.

10 responses



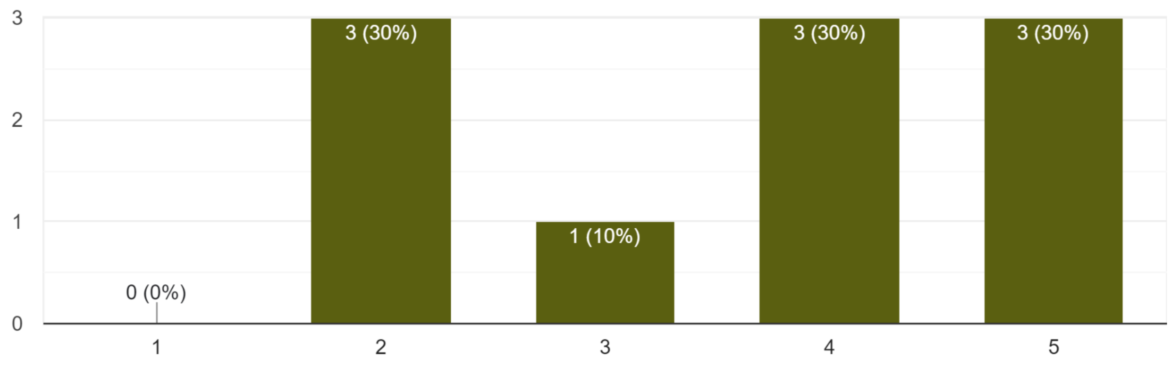
I found the system very cumbersome to use.

10 responses



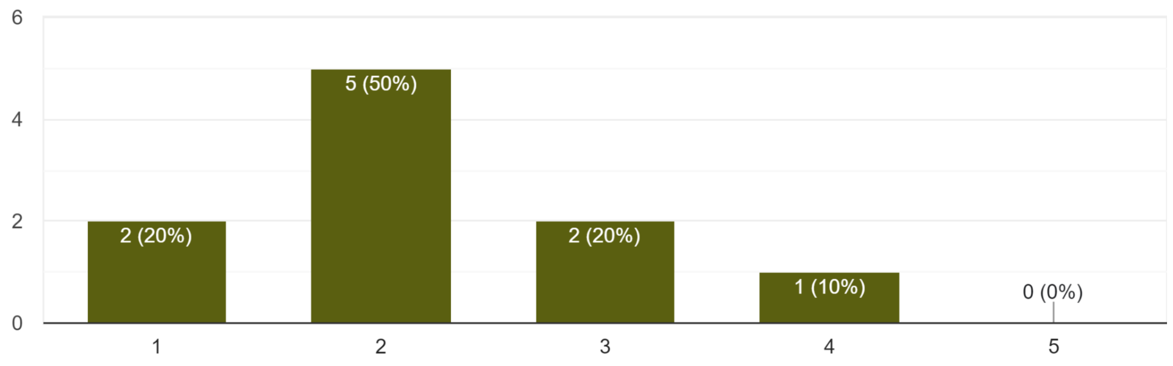
I felt very confident using the system.

10 responses



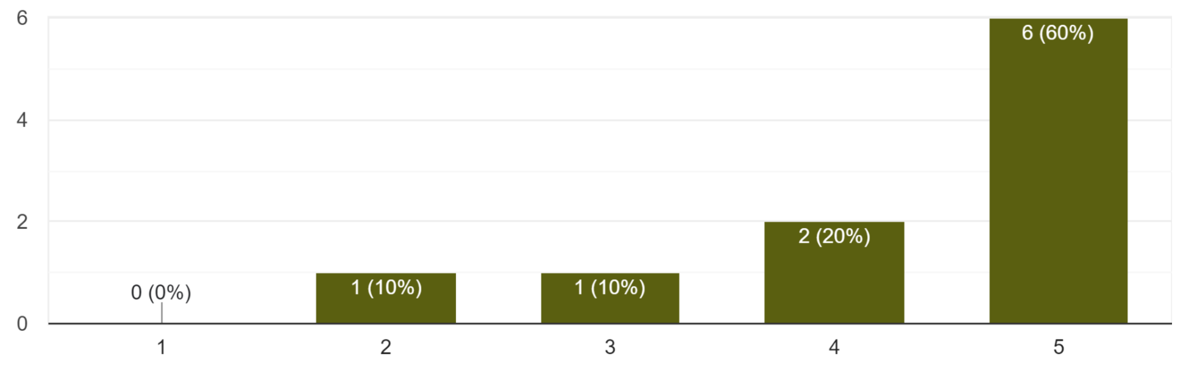
I needed to learn a lot of things before I could get going with this system.

10 responses



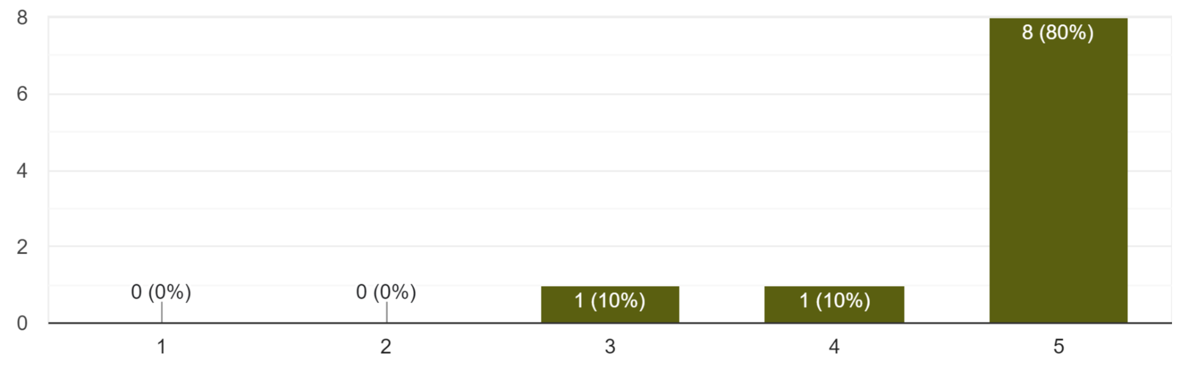
I thought the digital exhibits were well presented and positioned according to the QR codes.

10 responses



I think using a QR code is a good option.

10 responses



3. Interview Questions

Which design of the viewer App stood out to you the most?	Any thoughts with the Unity toolkit?	Which design of the entire system stood out to you the most?	What would you use this system for, if anything?	What do you think can be improved in this system, if anything?	More comments.
I liked the use of the qr code to create the AR experience, however the object appeared to be trapped on the same plane as the qr code.	I don't have extensive Unity experience, but the guide seemed to more than make up for the knowledge I lacked.	The use of multiple qr codes for different steps of the exhibit	I think it could be really helpful in teaching environments for things like understandin g molecules and other models.	Some clarity and streamlining on the process of creating objects.	A little more backgroun d before starting the playtest would provide clarity and meaning to the testers actions.

I liked the ability to hide the UI elements! I thought it made for a much better user experience when I was not attempting to use the buttons.	I personally do not have a ton of unity experiences , however, the instructions were very helpful and the pictures within it made for an easy process.	How easy it was to transfer the data from unity into a qr code was very impressive.	I would use this system if I was developing a piece of art (game assets or actual art) to allow for those to seemingly come to life in AR could be valuable to provide users with a unique experience.	I think that the button for recalibratin g should not be an X because it was not intuitive to look at and recognize what the function of the button was. I think that perhaps more pictures in the instruction could have been helpful, but honestly i did not need too much help.	I overall had a very positive user experience even as someone who is not familiar with Unity. I found the instructions a crucial and effective component of the process which helped me design something and bring it to life. I thought the app was very user friendly!

Interface	No	UI	To create AR exhibits or any other objects	No idea	
Using QR code.	None	QR code	I don't know.	It is good now	None.
Automatically loading the models through the qr codes was very fast and easy	The editor windows simplify the process for setting up the model a lot.	The cloud-based model storage made it easy to view any uploaded models	Sharing a 3D modeling portfolio	Interactive models could be very interesting	Having never developed AR technology, I was a little worried about complexity, but the system made the process very intuitive through unity and the custom editors.

It's very easy to use and the UI is very clean and good to understand.	It's very good to use with the short description. Users can use this toolkit without any AR knowledge.	The process to put model into empty gameobject is very quick and easy	If I create a model and want to share with others I will use this	Not much, maybe add some descriptions of uploading to cloud platform process	It's very easy to use and I think with this toolkit Ii can create AR projects very quickly.
simple UI	no	synthesize to cloud	exhibit products	It would be better to add more explanation for a user-friendl y purpose.	I like the UI~
Enable the state machine feature	You may add more "completed " objects in the test objects so that we can import the pre-defined objects with	Deploymen t to a light device (i.e. cellphone)	Quick visualization for a combined 3D scene	You may improve the perspective view directions so that you can rotate the scene intuitively	N/A

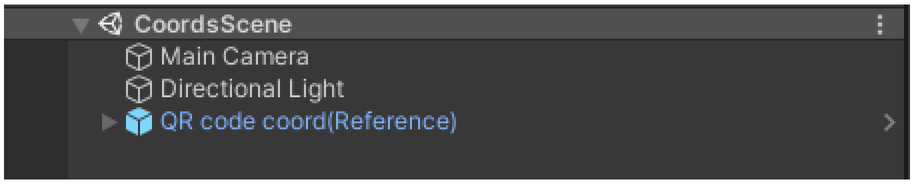
	more flexible options.				
It's pretty simplistic overall but that's not necessarily a bad thing.	It was pretty simple and the top of the screen is an excellent placement, for ease of access.	I was just so impressed to see how well it moved in "3D space" when I moved the camera across the QR code.	I'd use it to view my 3D models from Zbrush/Maya if I got them into Unity.	I wonder if there could be an easier way to generate the QR code, although I suspect replacing a generator would take a lot of work.	Nothing but praise, well done!
super simple to use	convenient	again yeah super simple and convenient	i think it would make ar games a lot easier to set up	easier way to set up single objects	

Appendix: AR MUSE Unity Toolkit Test Task

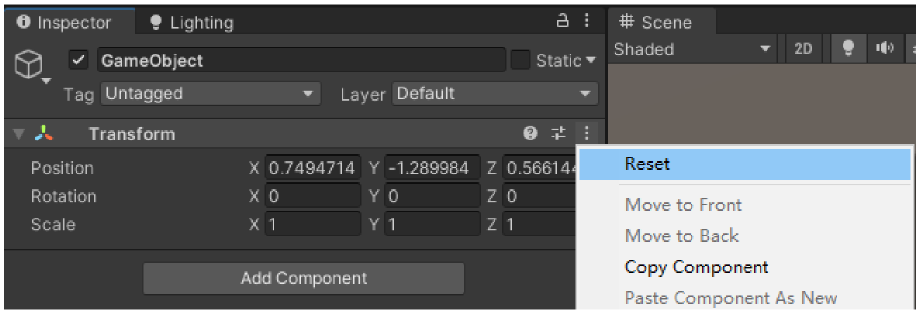
In this test, you are going to use Unity Editor to generate some AR-ready models, and then deploy them to a remote server in a few steps by using AR MUSE Unity Toolkit. In the end, you can use a mobile phone with AR MUSE Viewer App to have a look at what you have created.

STEP 1. Edit Model in Unity Editor with QR code Reference

1. Open **"Asset - Scene - CoordsScene"**



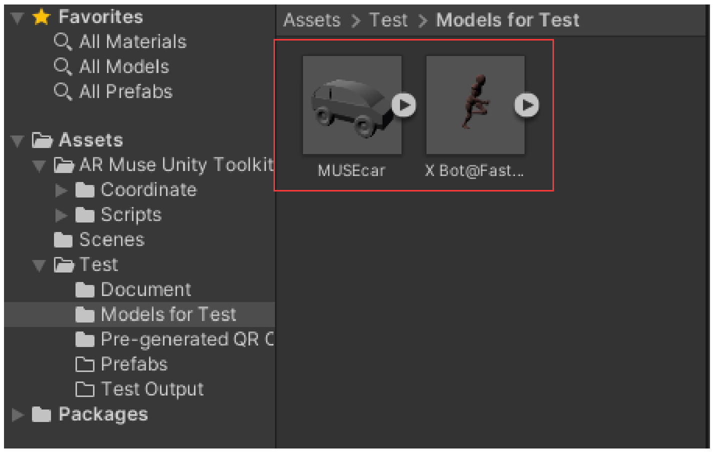
2. Press **"Ctrl + Shift + N"** to create an empty GameObject, this empty GameObject will be the container for your model. Then reset the **"Transform"** of the GameObject you have just created

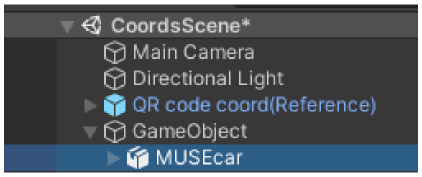


3. Find **"Assets - Test - Models for Test".** Select and drag one model to the empty GameObject created (This operation will make this model a child of the empty GameObject). Then you are free to make customizations to the models as you like (Position, Rotation, Scale…)

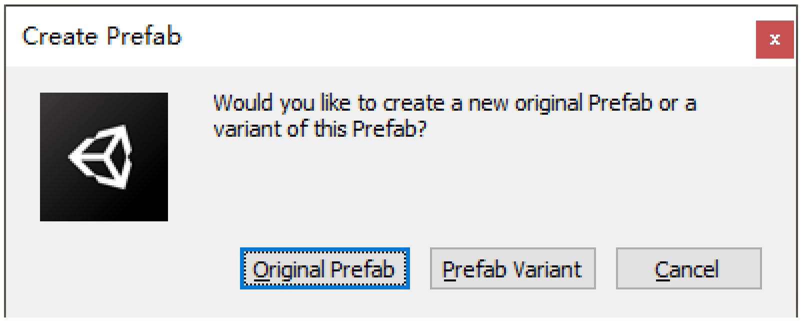
   Notice: *Any **"Transform"** changes made to the empty GameObject will be overwritten when using Viewer App, so instead of transforming empty GameObject, you **HAVE TO** transform the models, which is a child of the empty GameObject*

   Optional: *If you are familiar with Unity, the **"X Bot"** model comes with an animation clip, that means you can animate it by adding the **"Animation"** or **"Animator"** component*





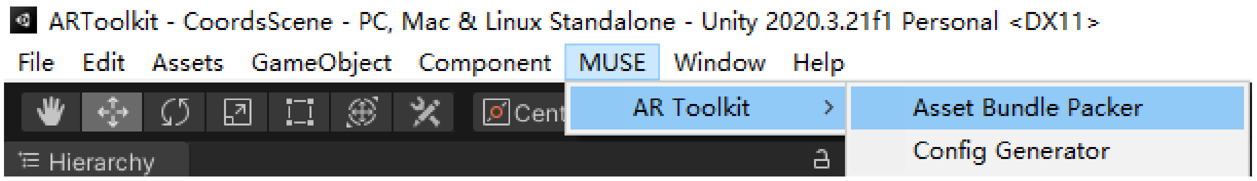
4. Select your model and make a new prefab by dragging it to **"Assets – Test - Prefabs"**, then select **"Original Prefab"** to save your customized model as a prefab


Create Prefab

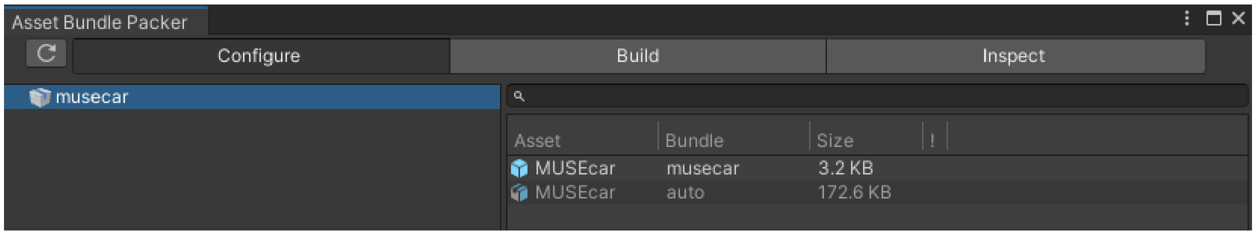
Would you like to create a new original Prefab or a variant of this Prefab?

STEP 2. Pack Asset

1. Select **"MUSE"** on the top menu bar, go to **"MUSE - AR Toolkit - Asset Bundle Packer"**

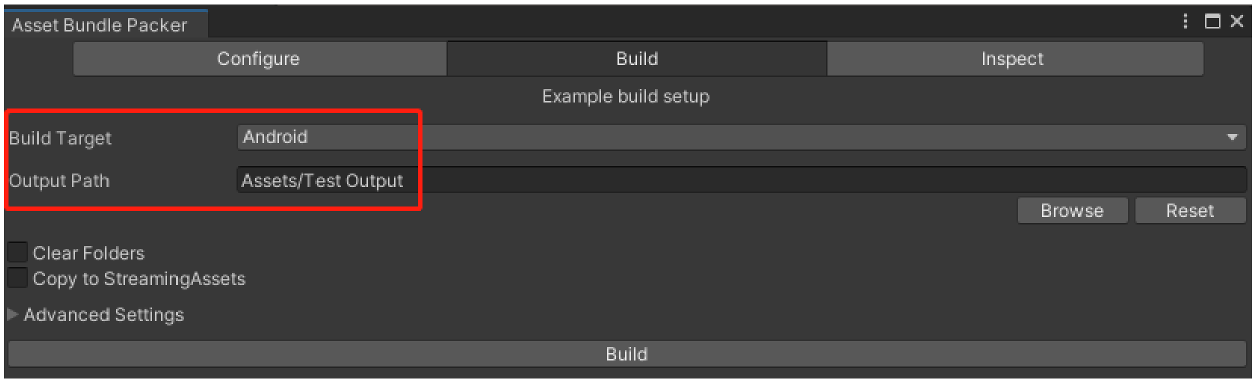


2. Drag your prefab to **"Configure"**



Asset	Bundle	Size	!
MUSEcar	musecar	3.2 KB	
MUSEcar	auto	172.6 KB	

3. Select **"Build",** and choose **"Build Target"** as **"Android",** and we have set the output path for you



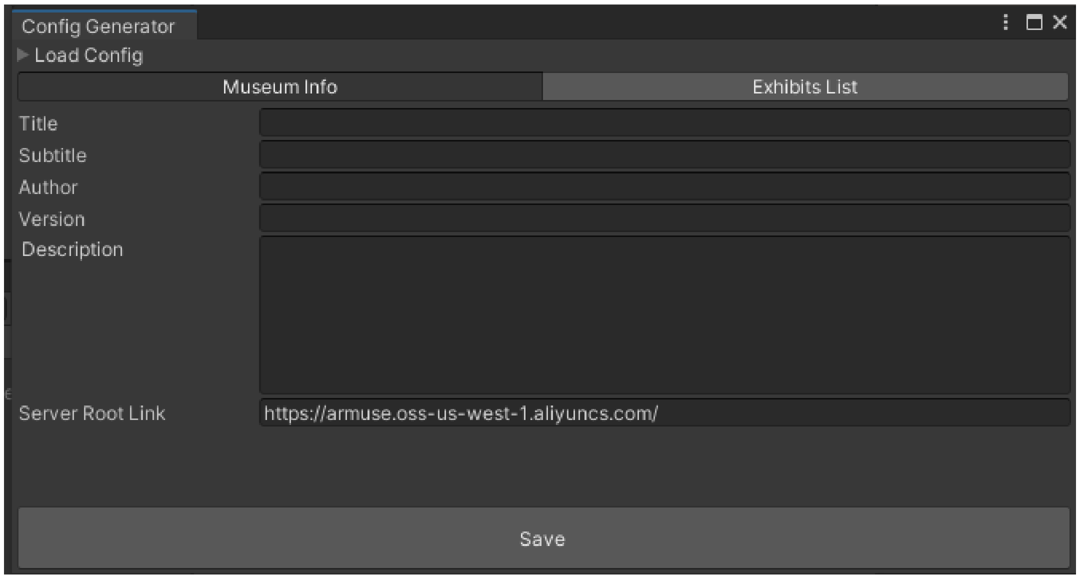
Build Target: Android
Output Path: Assets/Test Output

4. Click **"Build"** button to pack your assets

STEP 3. Generate XML Config File

1. Select **"MUSE"** on the top menu bar, go to **"MUSE - AR Toolkit - Config Generator"**

2. Select **"Museum Info"** to edit your AR museum/exhibition Info


Config Generator
Load Config
Museum Info | Exhibits List

Title
Subtitle
Author
Version
Description
Server Root Link: https://armuse.oss-us-west-1.aliyuncs.com/

Save

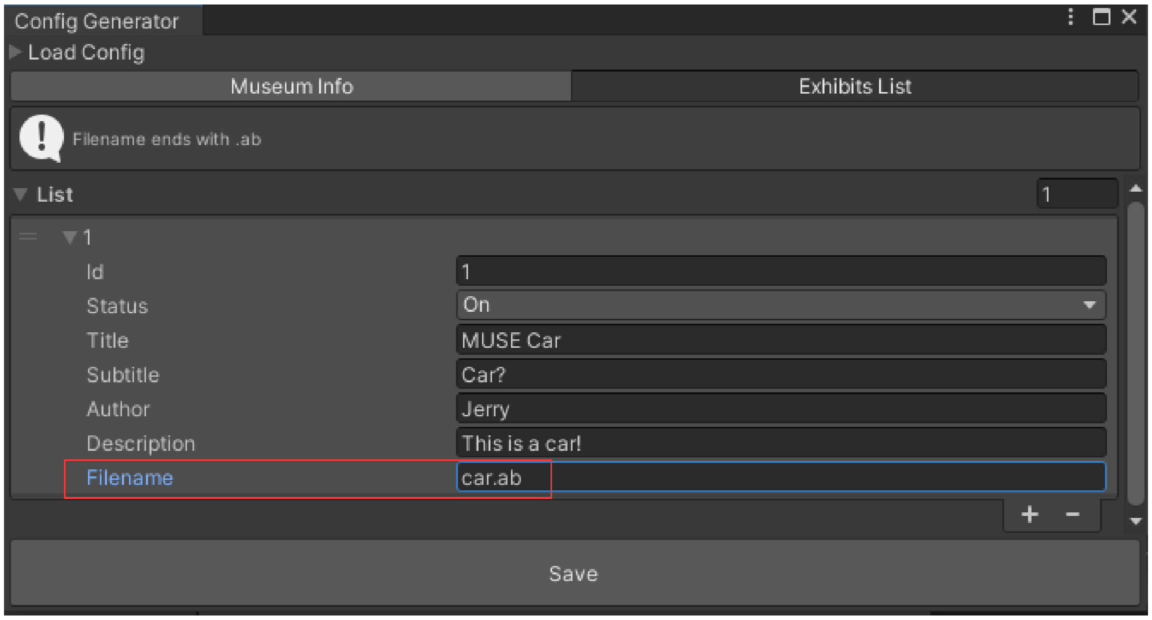
3. Check if **"Server Root Link"** equals **"https://armuse.oss-us-west-1.aliyuncs.com/"**, if not paste it into text field

4. Select **"Exhibits List"** to add new Exhibits info, you can add multiple exhibits into this list

   *Make sure you have input **ALL** the information*

   *Make sure the **"Filename"** of each exhibit is the same as the packed asset name and ends with ".ab".*

   For example: car.ab


Config Generator
Load Config
Museum Info | Exhibits List

Filename ends with .ab

List: 1

1
Id: 1
Status: On
Title: MUSE Car
Subtitle: Car?
Author: Jerry
Description: This is a car!
Filename: car.ab

Save

5. Click **“Save”** button to save your config file

STEP 4. Upload To Server and Generate QR code

Let us do this for you :)

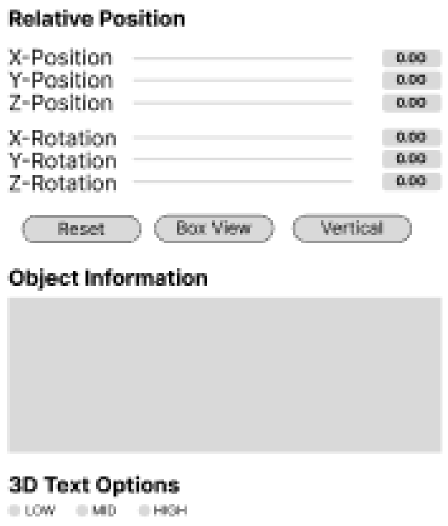
STEP 5. See it on Viewer App

1. Scan the Museum Entry Code
2. Scan the Exhibit Code and see your uploaded models

   Optional: You can interact with any buttons you see on the screen, feel free to play with the Viewer App. We will be happy if you find some bugs or problems ;)

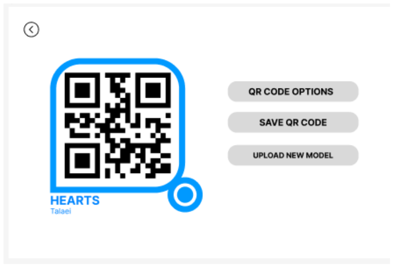
Appendix: Detailed Information of AR MUSE

1. Curator App



Curator App has been designed to be a simple tool for anyone without the related Unity still . Therefore, the interface was very simple, with only three interfaces:

- Main page
  It's just a welcome cover that doesn't contain any real functionality.
- Editing page
  Users can input their model file and make simple edits to the model here. In the original design, for maximum users' convenience. The input model will be auto adjusted to fit the size of QR code. For example if the QR code size is 10x10. A 80x80x100 model will be scaled down to 8x8x10.
  Specifically includes functions:
  - Translation based on XYZ axis
  - Rotation based on XYZ axis
  - Add a text pad in front of the model. User can choose from 3 default positions: Low, Mid and High
- Result page
  After finishing editing. The user can click upload. Based on the original design, this file will be uploaded to the server and generate a QR code that is bound to this uploaded model. Users can now save this QR code and share with others.



2. Unity Toolkit

This AR MUSE Unity Toolkit is a package of Unity Engine. Users can set it up by downloading from the official Asset Store, and then import to an empty project. This toolkit is an alternative to selecting assets and setting their asset bundle manually in the inspector, as well as hand-writing a xml config file. This toolkit can be dropped into any Unity project with a version of 2020.x or greater. It would create a new menu in MUSE > Unity Toolkit > Asset Bundle Packer and MUSE > Unity Toolkit > Config Generator.
Asset Bundle Packer is a powerful tool that provides a window, and enables the user to view and edit the configuration of asset bundles for their Unity project. It will block editing that would create invalid bundles, and inform the user of any issues with existing bundles. It also provides basic build functionality. The bundle configuration, build functionality, and build-bundle inspection are split into three tabs within the window.

- Configure: This window provides a Windows file explorer like interface to managing and modifying asset bundles in a project. When first opened, the tool will parse all bundle data in the background, slowly marking warnings or errors it detects. It does what it can to stay in sync with the project, but cannot always be aware of activity outside the tool. To force a quick pass at error detection, or to update the tool with changes made externally, the Refresh button in the upper left is designed to solve this. When a user drags assets into this window, this tool will automatically detect the dependency of the asset.
  Configure window is broken into four sections:
  - Bundle List, which is a left hand panel that holds a list used to browse prefabs needed to be packed into asset bundles. If a bundle has any error, warning, or info message, an icon will appear on the right side. Users can mouse over the icon for more information. In particular, if a bundle has at least one scene in it, making it a scene bundle, it will be marked as having an error. This bundle will not build until all scenes are removed. Bundles with duplicated assets will be marked with a warning, more information on duplication is available in Asset List we are going to introduce below.

  - Bundle Details, which is a lower left panel indicating file size (a sum of the on-disk size) of the currently selected bundle, and its dependency (bundles that the current bundle depends on). Also, this panel gives user warning, error, or info messages associated with the currently selected bundle.
  - Asset List, which is an upper right panel providing a list contained in whichever bundles are selected in the Bundle List. The search field above this list will match with assets in any bundle. The Asset List will only display matching assets, and the Bundle List mentioned above will only display bundles that contain matching assets.
  - Asset Details is a non-interactive lower right panel. It is similar to Bundle Details, but instead of showing details of the bundle, it only displays the full path of the asset, the reason for implicit inclusion in bundles if it is implicit, and the reason for error, or warning if any.
- Build: The Build tab provides basic build functionality to get you started using asset bundles. In most professional scenarios, users will end up needing a more advanced build setup, and thus need some coding and modifying, but when considering this toolkit's target user is the artist or the curator with limited computer knowledge, we moved most of the cumbersome features into advanced settings to minimize difficulty of using this toolkit, like compression type, append hash, and strict mode. The basic build options contains:
  - Build Target: Platform the bundles will be built for. Due to Unity Engine's restriction, asset bundles can not work cross-platform, so the user needs to select a platform before build.
  - Output Path: The file path for saving built bundles. By default this is AssetBundles/. The user can edit the path manually, or by selecting "Browse". Hit "Reset" can return to the default naming convention.
  - Clear Folders: When toggling this option, all data from the build path folder will be wiped completely prior to building.
  - Copy to StreamingAssets: StreamingAssets is a persistent path that Unity

holds, when toggling this option, toolkit will automatically create the StreamingAssets folder if it does not exist, and copy results to it. l

- Inspect: This tab enables the user to inspect the contents of bundles that have already been built. Add file function enables the user to view a single asset bundle, and Add Folder gives the user ability to inspect all asset bundles belonging to that folder. When inspecting multiple asset bundles or adding a folder, Inspect tab uses the tree view struct that is similar to Unity's original Hierarchy window. Also, this tool will send Console Warning or Error when the user is trying to inspect an invalid or broken asset bundle, this could help them find problems in early stages.

Config Generator is a tool that allows the user edit museum's information, as well as each model's, all the information will be written into a xml file, which will be uploaded to the server later manually. This tool will detect the validation of each information before writing it into a xml file. It supports loading a previously generated config, which is necessary when the curator or artist is trying to maintain and update an AR museum.

- Load Config: This function allows the user to read the config file generated before. Only file with xml format is supported, and the program will automatically check if this xml file's format is the same as set by comparing its hashset. Data will only be loaded and displayed after passing this detection.
- Museum Info: This tab is used to edit museum's information. Museum information contains:
    - Title, which is the name of this AR museum.
    - Subtitle, which can be a detailed title that explains this AR museum's title.
    - Author/curator, which is the owner, curator, or sponsor of this AR museum.
    - Version, this is used for version control and hot update.
    - Description, which can be an explanation or background of this AR museum.
    - Server root link directed to this museum, this is used for telling Viewer App where to find these packed models when scanned a QR code.

- Exhibits List: This tab displays a list of the exhibits' information. The usage of Unity's default list view is an easy and intuitive way to manage all the exhibits. Exhibit's information consists of:
  - Id, which is just an identifier for recognizing and managing, which means Viewer App do not use this as the identity of each model.
  - Status, this parameter tells Viewer App whether this model is on display, or not. It has only two sections: On or Off. Only models with On status can be scanned and are available in the Collection List.
  - Title.
  - Subtitle.
  - Author.
  - Description.
  - Filename, Viewer App uses this to recognize each QR code, in other words, this is the identity of each model. Filename must be the same as the packed models' name saved on the server, and ends with ".ab".
- Save: This button is used for saving and writing all the information into a xml file. However, this tool will do a hashset detection before saving, if there's any incompletion or any filename does not end with ".ab", it will give an error message.

3. Viewer App

The Viewer App is a mobile application used for exhibition audiences to scan QR codes and then download pre-packed models to their phones, and then Viewer App will unzip the packed models, and display them in the AR environment. In other words, this Viewer App is a combination of QR scanner, reader, and also provides a platform to render these models in an efficient way based on Unity's AR Foundation package. As an app designed for the AR museum, this app offers a scene that displays all the exhibits on showing. This viewer app consists five scenes:

- Opening: This scene is the beginning of this app, exhibition audiences can hit the enter button to start their journey.
- Entry: This scene is used for the exhibition audience to scan the entry QR code. When first entering this scene, this app will automatically clear all cache, and persistent data (playerprefs) saved before in order to overwrite the new scanned config date. Since there is not much complicated scan and read work in this scene, in order to make the scan progress less time-consuming (because most of the museum's entrance is crowded), we decided to use a simplified state machine compared with the state machine used in the main scene, which will be discussed next. This simplified state has five states:
  - Scanning: This is the root state, and only appears when this app keeps scanning images from the camera.
  - Detected: When scanned a QR code, the current state will be switched to detected. This will also activate a dialog that tells the user there is a QR code detected, and try to read data from it. Simultaneously, this app will start checking the QR code's content validity, this app will start downloading the config file from the server when it passes detection, or it will return to scanning state.
  - Downloading: This state will only be triggered when this app scans a QR code, and the content is not empty and ends with "xml". We use UnityWebRequest function to download from the server, if the downloading process fails, the dialog will notify the exhibition audience. Once downloading is finished, the state machine will move forward to the next.
  - Analyzing: Contains a series of operations in this state. Like the hashset detection mentioned in Config Generator part, firstly the downloaded data will be transformed into a xml file, and check each element's name and innertext's validity. If any part fails this test, the state machine will return to the root state, and this app will give the exhibition audience a "Invalid config file" error.

- Done: After finishing analyzing the downloaded file, the app will write the entire xml file into cache, and allow exhibition audiences to enter the museum.

- MainScene: This is where exhibition audiences scan QR codes and play around with models. This scene is divided into three sections:
  - Toolbar: Lies upper middle that has a menu button, which contains entrances for About scene, Collection List scene, and an app exit, right side of the toolbar is a button for hiding UI except itself, this could help exhibition audiences better observing models and increase immersive feelings.
  - Body Focus Frame: It is an indicator that tells the exhibition audience the right size of the QR code. Additionally, in this part we will talk more about the state machine used to control QR code tracking and scanning progress. In order to get better performance and positioning accuracy, the scanning state and detected state we used in simplified state machine is broken into four steps:
    - Searching: Just like the simplified one, this is the root state of the scanning state, as well as the entire state machine.
    - Stabilizing: When finding a QR code, this app will not try to read content from that code until the tourist's mobile phone is stable enough. Only a tourist holding his phone longer than the stabilizing time set by the curator can switch to the next state.
    - Registered: In this state, this app will try to download models from the server like the operations we mentioned at simplified one, validity check is necessary here. Also, once QR code passes the check, this app instantiates an empty object at the position of that QR code, this is used to be replaced by the models, and sends a download request to fetch models from the server. The download process is the same as the simplified one.

■ Refine: This is specifically designed for the phenomenon that in an AR museum, after scanning a model, exhibition audiences will usually walk around, take a look, and move their phones, which will lead to inaccurate positioning. Refine step is a loop that keeps updating QR code's position in the real-world, and when it loses track for some time, unregisters that QR code, and destroys instantiated models, in the end, returning to root state.

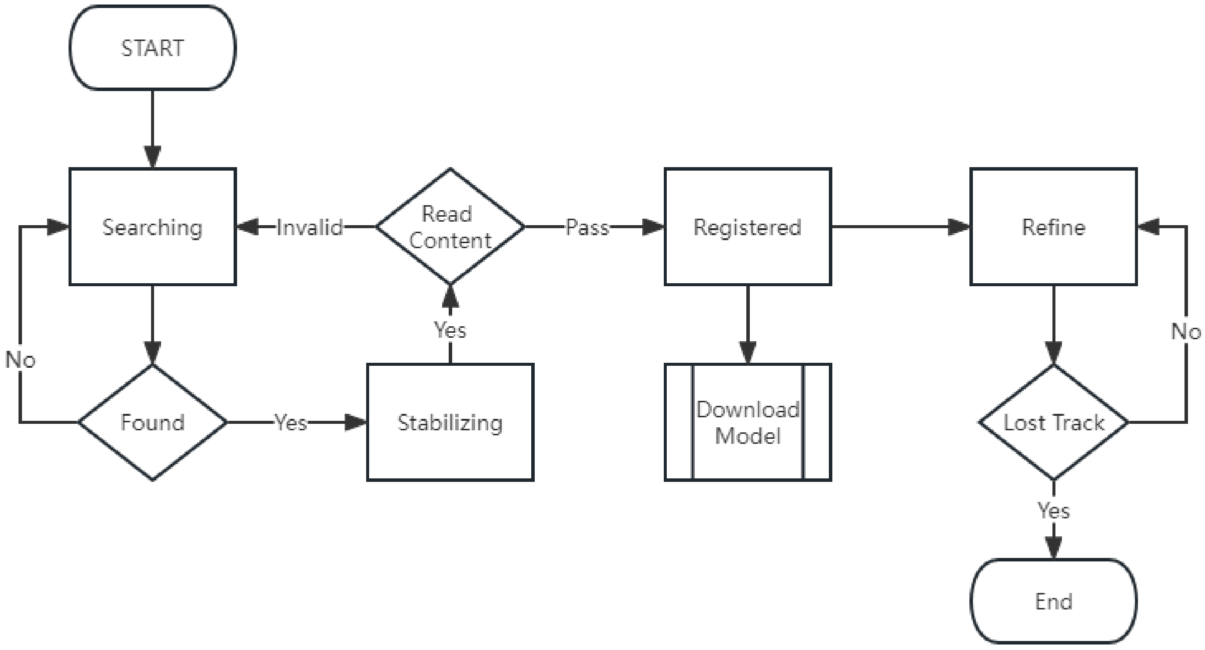


Figure: QR Code Scan Process

○ Bottombar: When exhibition audiences first enter this scene or when there is no model on display, the bottom bar shows the museum's title and subtitle. In normal cases, this bottom bar is folded to hide the detailed description, only when the exhibition audiences hit the expand button, this bottom bar will unfold to display all the information. Additionally, there is a clear button to unregister the QR code manually, which will also destroy the instantiated model as explained before.

- About: This is a simple scene that is used by the curator to write anything about the museum, or it can be a tutorial for exhibition audiences.
- CollectionList: Here provides a list of currently on showing exhibits. Exhibition audiences can click each single exhibit, and inspect that exhibit's description. We believe this could help exhibition audiences decide whether to have a look at that model, especially when they are in a large scale museum.