V-ART: an Augmented Reality Museum Companion

Albert Sandru Andrei Cotor Berndt Uhlig William Jones
ETH Zürich, Switzerland
asandru@ethz.ch acotor@ethz.ch buhlig@ethz.ch wjones@ethz.ch

Abstract

Recent advances in generative AI have allowed for high-fidelity 3D worlds to be generated from a single image, with Gaussian splatting becoming increasingly popular due to its flexibility and applicability to the output of neural networks, being more lightweight and easier to generate than traditional meshes or NERFs. Experimentation with Gaussian splatting has begun in the worlds of virtual and augmented reality, with the intuitive idea of viewing these 3D worlds in real space. With the advent of Gaussian splats, rendering libraries were developed for Unity and SparkJS, with a small community forming around each.

We present V-ART, a museum companion app developed for the Meta Quest 3 in Unity. Our app allows museums to combine traditional painting information with novel Gaussian splat representations, with an integrated QR-Code scanner to provide contextual interaction possibilities. Users can navigate their surroundings, locate a painting and load the associated "painting world", physically exploring it as they go. The user response to our application was highly positive, with test subjects praising both immersion and the general look and feel of the app.

1. Introduction

Generative AI has seen a huge surge in interest as the quality of its results increased with advances made over the past years. While a large amount of attention is placed on video and image generation, high quality 3D generation has also seen remarkable progress. One of the major strides was made possible by a technique called Gaussian splatting, which exploded in popularity starting in 2023, allowing for accurate 3D novel view synthesis and representation. Gaussian splats are fundamentally different from traditional 3D meshes and polygon rendering, meaning that their usage for various areas of application is dependent on the existence of rendering pipelines for established tools. For the Unity game engine[6], the current most popular solution is the UnityGaussianSplatting repository [5] by aras-p, which utilizes custom shaders to effectively render Gaussian splats.

The project started with the simple idea of visualizing 3D worlds created with Gaussian splatting in a Virtual Reality headset. From here, we developed the idea of exploring the worlds depicted in famous works of art, something we envisioned fitting seamlessly into a museum environment. With this basic concept, we performed an initial user study to collect feedback on what features users would like to see in such an app. We aimed to see if there was any interest in the concept, as well as gathering opinions on what users preferred: a fully virtual museum that users could explore from the comfort of their own home, or an augmented reality museum companion app that would improve a real-life museum-going experience. Participants overwhelmingly opted for an augmented reality companion application equipped with features such as museum navigation and painting information, as well as the possibility to view the 3D Gaussian splat worlds.

V-ART, therefore, is designed to act as an AR museum companion, allowing users to explore the museum freely and uninhibited, while having the options to view the museum map, collection of paintings, and immersive 3D worlds right at their fingertips.

2. Related Works

2.1. Gaussian Splatting

Gaussian splatting is a volume rendering technique, which was originally introduced in 1991 by Lee Alan Westover [7]. With advances in AI, it resurfaced in 2023 [2] as a scene representation that lends itself to generation by neural networks and has enjoyed considerable attention and popularity since. Traditional 3D scene representations use polygonal meshes, voxels or distance fields, which are then rendered. Gaussian splats are fundamentally different. Each particle is a 3D ellipsoid, with a position, rotation, and non-uniform scale in 3D space. Particles also store color and opacity information. The color is not represented as a single value (i.e. an RGB value), but rather as third order Spherical Harmonic coefficients, allowing the color to change with view direction. The name "splatting" comes from what happens during rendering: Each Gaussian is placed sequen-

tially into the scene, like paint being splatted onto a canvas to create the full image. That is to say, each 3D Gaussian is sequentially rendered in screen space as a 2D Gaussian. This approach is considerably simpler than comparable approaches like NERFs. Solutions have been found to train AI models to represent scenes depicted in multiple photos (or a panoramic image) as splats, paired with fast and efficient rendering algorithms that run on GPUs.

2.2. Architectures

During our testing, we investigated the following models for 3D world creation:

- **WonderWorld**[10]: *Interactive 3D Scene Generation from a Single Image*
- **CubeDiff**[1]: *Repurposing Diffusion-Based Image Models for Panorama Generation*
- **Matrix-3D**[9]: *Omnidirectional Explorable 3D World Generation*
- **Marble**[8]: *A Multimodal World Model*

WonderWorld was the initial paper that we intended to use for our app. The WonderWorld model allows for real-time scene generation with an initial world being created based on an input image, which can then subsequently be explored and "colored in". This is achieved using guided depth diffusion and a highly optimized "FLAGS" (Fast LAyered Gaussian Surfels) representation. The model generates an initial scene, and then adds onto it as the user moves the camera, or requests specific changes. This setup allows for real time generation, meaning as a user explores the world, new areas are dynamically generated. The main drawbacks, however, are that for this to actually function within anything resembling real-time, it requires very powerful hardware (the paper suggests an A6000 GPU). Additionally, the FLAGS representation produces a much shallower 3D world, meaning it is dependent on its ability to generate new parts of the scene dynamically to maintain visual fidelity. While significant time was spent trying to adapt this model to fit our needs, we elected to search for other options, as the hardware limitations of WonderWorld were too restrictive, and non-interactive world generation produced less than ideal results.

CubeDiff is a novel method for generating 360° panoramas from text prompts or images. This is achieved using multi-view diffusion models to jointly synthesize the six faces of the cubemap, with each face being treated as a standard perspective image, which simplifies the generation process. A prospective user has a high degree of control over the generation process, as they can provide text prompts for every face of the resulting cubemap. We investigated its use for our app to see if a 360° panorama image could be an effective replacement for a 3D environment. In this scenario, we would use the image as a form of skybox. This would be highly performant, and due to the high speed of CubeDiff, generation could also be done "on the fly". This, however, comes at the cost of 3D exploration. The result is simply a 360° panoramic image. When used as a skybox, the picture is always projected at the horizon. As the image is two dimensional, it lacks depth and will naturally also "follow" the user as they move. We were ultimately unhappy with the results, especially as 3D exploration was a core part of our initial plan. Part of the appeal of VR technology is the reaction of the world to head movements, the simulation of depth and perspective, which would simply not be present in this case.

Matrix-3D generates a 3D world via first generating a trajectory guided panorama video and then using the resulting video to generate a 3D world. It takes an initial image as input, as well as a trajectory through which the camera "travels", representative of the area which a user would like to explore. It then generates a panoramic video, which is turned into a 3D scene using a "large panorama reconstruction model". The result is a 3D Gaussian splat representation with remarkably high quality and visual fidelity. However, while this model initially seemed promising, our hardware was too limited to run it at all, as none of the GPUs available to us possessed the necessary VRAM requirements. Even with their "Low-VRAM" mode, it requires 12GB of VRAM, which was inaccessible to us.

Marble is the current state-of-the-art model by World Labs. Its a generative multimodal world model, allowing 3D world generation from text, images, panoramas, videos, and coarse 3D representations. Worlds can be shaped, edited, and combined, with the final result being exported as Gaussian splats. It is currently accessed through a paid online service. It provides highly consistent results that can further be fine-tuned as desired, generating immersive and fully explorable worlds. This is achieved in a multi-step process, first generating a panoramic image around an initial input, which is then used for depth estimation. This information is used to transform the scene into Gaussian splats. Each step of this process allows for user input, therefore the intermediate results can be fine-tuned and altered. Our initial testing showed that Marble delivered consistent results when paintings are used as input, which, due to the nature of different art styles can be quite challenging for steps such as depth estimation. Furthermore, the versatility of the associated online tool provided by WorldLabs allowed us to quickly create various iterations of desired worlds and fine-tune the results. Due to its performance, we elected to utilize Marble as the Gaussian splat generation model for the final version of V-ART.

3. Methodology

V-ART is built for the Meta Quest 3 headset using the Unity game engine. The app was designed to incorporate features

and conveniences that users have come to expect in a traditional museum-going experience (e. g. background information on the artwork, a catalog overview, a museum map) with novel, augmented reality-enabled features such as immersive 3D environments inspired by the art. In this section, we will explore the technologies involved in creating the app and will explain the key points of the app's architecture and design.

3.1. Application Framework

For the best compatibility, we created **V-ART** in Unity version 6000.0.62f1 with Meta's Mixed Reality Utility Kit (MRUK) [3]. This decision evolved with the project, as originally the suggested framework was a Unity 2022.3 version in combination with Microsoft's Mixed Reality Toolkit (MRTK) [4]. While MRTK is a viable option for, and indeed works well with, many AR applications, we needed access to newer features of Meta's newest headset, the Quest 3, which MRTK simply did not have access to; Meta's own mixed reality SDK exclusively gives developers access to the entire breadth of the headset's capabilities. The project was updated to Unity 6 as the library used for displaying Gaussian splats was incompatible with Unity's universal render pipeline (which is a MRTK and MRUK requirement) for any version under Unity 6.

3.2. Gaussian Splatting Framework

To allow users to jump into the worlds that a museum's artworks present, we first elected to generate our application's Gaussian splats using WonderWorld. WonderWorld allows users to create an explorable 3D scene from a single image interactively and in real-time. Scene creation is done via a browser tab where users can "explore" a given image in which new content is created on the fly. Additionally, creation can be steered by providing WonderWorld with text prompts to specify scene contents. Unfortunately, our experience with WonderWorld did not align with their real-time promises. Real-time generation was only possible with an NVIDIA A6000 GPU which we neither have nor were provided with. This meant that a single panorama took upwards of 24 hours to create on an NVIDA RTX 4060. This means that the intended generation technique was also not usable, as each input took about 40 minutes to materialize.

To avoid this limitation, the code was adapted to automatically provide input by rotating the virtual camera once the new content of the scene finished generating. Using this method, we could specify an input image and return the next day to retrieve the Gaussian splat. Unfortunately, output scenes were not always of good quality, showing strange tearing and AI artifacts where the depth or content of the scene was not understood, resulting in another 24-hour wait for a new scene. Additionally, WonderWorld generates dense splat skyboxes that resulted in bloated splat counts that did not perform well in the final program. To this end, the code was also adapted to prevent WonderWorld from generating these skyboxes at all.

Though the aforementioned changes to WonderWorld made it theoretically possible to create Gaussian splats, we elected to find another framework. This led us to Marble. Marble was able to do what WonderWorld could not and allowed us to quickly create splats based on our input artworks. Due to its speed and in-built editor tools, we were also able to edit and steer the creation of the splats until we were left with a result fit for our application. Marble creates splats in two sizes, small and large, with 500 thousand splats and 2 million splats, respectively.

3.3. Gaussian Splatting Visualization in Unity

Gaussian splats do not yet work natively in Unity. Fortunately, as Gaussian splatting has been garnering more attention, there exists an open-source library, UnityGaussianSplatting, which, as the name suggests, allows for Gaussian splat rendering in Unity. Unfortunately, the implementation does not work out of the box for the Quest headset. Without any tinkering, splats are visible only in one eye, are always displayed upside down, and exhibit unusable performance, even with a very low number of splats.

We attempted to create our own Gaussian splat viewer to see if a simpler approach – not using complex custom shaders that conflicted with VR headsets – would work. A first approach in which we simply read the splat file and instantiate a sphere object for each splat, stretched to match the shape, was not usable. Because each object creates its own instantiated mesh requiring its own draw call, which must update the CPU render state and then submit the information to the GPU, the performance hovered around 0 frames per second. However, individual draw calls can be solved with GPU instancing. As each splat actually has the same mesh – a sphere – the draw call can be optimized, and the GPU is only sent the information of one mesh which is then applied to an array of objects in the scene, each of which can still have its own color or scale. Unity's native GPU instancing function only allows for 1023 objects to be passed to a single call. As our environments contain 500 thousand splats, this method still proved to be too slow, so we returned to the UnityGaussianSplatting method.

UnityGaussianSplatting can, in fact, work on VR headsets. We were able to solve the aforementioned issues with several fixes:

- Enable multi-pass rendering in the graphics settings, performing a render pass for each eye separately, preventing the need for complex headset shaders. Performance is not affected.
- In the render settings, ensure that HDR is not off while also having Intermediate Texture setting set to "Auto" – the combination of these two settings results in upside-

down rendering of splats
- Set the UnityGaussianSplatting "Sort Nth Frame" setting to a higher value — this controls how often the splats are reordered and, when done too regularly, accounts for the greatest performance loss

With these changes, the Meta Quest 3 was able to render Gaussian splats at a usable performance – see section 4.3 for specifics.

3.4. The Painting Object

As **V-ART** is not only a Gaussian splat viewer, we collect more information on each viewable painting and store and process it internally in a C# **Painting** class. Each Painting instance contains a unique identification number, references to its actual image, coordinates of its real location within the museum, as well as its Gaussian splat representation. Additionally, it stores information that a user would expect to see in a museum environment: its title, artist, medium, and a description of the painting. This internal representation allows us to easily update and change a painting's data, for ease of use when "curating" a new museum. The Painting ID is used to identify it with QR codes placed in the museum, so that when a user approaches an artwork, the system can recognize which piece they are viewing and provide relevant information and actions. A Painting stores its coordinates so that the user can choose an artwork from the provided list and have its location displayed on a virtual map. Further, we added the ability to assign an audio guide to a given painting. An audio source component is linked to the painting, which can then be played, paused, or restarted. The audio stops playing when the painting information panel is closed, but continues playing should a user decide to open a paintings respective panorama, allowing the experience to be further enhanced. It should be noted that this functionality isn't limited to plain audio guides. It could feature music, environmental noises befitting the scene or actual little narratives for entertainment purposes.

3.5. QR Codes

QR code recognition also accounted for several implementation challenges. MRTK, our original augmented reality framework, did not allow for access to the Quest's camera feed, a crucial feature for detecting QR codes. We attempted to get around this limitation by allowing the user to "opt-in" to QR code scanning by taking a screenshot of the QR code and reading the QR code from the saved image. The Meta Quest 3 is very protective of it's internal memory, however, blocking all attempts to retrieve the screenshot, so this approach left us empty-handed. Only after switching to MRUK were we able to make headway – Meta allowed MRUK users to access the passthrough camera feed as of March 2025. With this, we are able to retrieve images the user sees and scan each image for a QR code. QR codes are read using a .Net implementation of ZXing. ZXing has tools for the extraction of QR codes from images and allows us to easily retrieve their data. In our application, the QR codes simply encoded each Painting's unique ID so that when read, a contextual menu appears.

3.6. High-Level Design

With the frameworks and functionality in place, the design of **V-ART** was created to function as follows:

1. The application opens to a brief tutorial screen instructing users how to interact with the application.
2. A hand gesture opens the main menu, closing any other open menus. From this menu, users can open the list of available paintings, a museum map, or the tutorial screen again.
3. The painting list is a list that shows the paintings on display. It can be scrolled through, and tapping on a painting opens an information screen for that painting.
4. The painting information screen contains information that would typically be found in a museum, including title, artist, medium, and description. From here, the user can play an audio recording presenting them with more information on the painting. Additionally, the user can tap on a button to pinpoint the location of the painting on the map.
5. The map menu displays a map of the environment and has a 3D pin model to indicate the location of the selected painting. Interactive navigation is unfortunately outside the scope of this project.
6. When a user arrives at a painting, an adjacent QR code is automatically scanned, prompting the user to open the painting information menu, providing the user with the same information menu accessible through the painting list with the new addition of a button that virtually places the user within a corresponding Gaussian splat world based on the painting.
7. From within a Gaussian splat, the user can perform the same menu hand gesture and close the view, opening the menu to continue exploration.

The option to view Gaussian splats only when physically in front of the painting is intentionally only accessible from the menu opened via the QR code, as we want to encourage the user to interact with the real artworks and systems of the application. Additionally, as the user will not be able to see their real-life surroundings while viewing gaussian splats, this is intended as a safety feature so museums can design their exhibits with this use in mind, keeping freely-wandering splat explorers to a minimum.

The final application is designed to be used without controllers, as we wanted to minimize the deviation from a normal museum-going experience and reduce necessary equipment. Interaction with the entire application is done via hand controls and gestures.

4. Results

4.1. Final Product

Figure 2 depicts the user interface of the painting collection. This is a scrollable menu that resides in world space that allows users to view at a glance the list of paintings that are available within the application. Tapping on an item in the list brings the user to the painting information menu from which they can learn more about the painting.

The painting information menu can be seen in figure 3. This menu lists information on the chosen painting and allows the user to view the paintings location on the map and either view a Gaussian splat of the painting or return to the painting collection menu, depending on how the user navigated to the menu.

Figure 4 shows the in-app map. The map menu is tied to the user's left hand, following it as it moves. This allows the user handle the virtual map as if it were an analog map, simply holding the map up to eye-level to view it, and lowering their hand when they no longer wish to. If the user selected to view a painting's location, a 3D pin model is placed on the map at the painting's location.

4.2. User Study (Demo Evaluation)

We conducted a user study to evaluate the demo version of our application in terms of perceived usefulness, visual quality, usability, and overall user experience. The study was designed as a post-use questionnaire-based evaluation, combining quantitative Likert-scale ratings with optional qualitative feedback.

Participants. A total of 16 participants took part in the study. Participants included Bachelor, Master, PhD students as well as industry professionals. Self-reported prior experience with AR/MR systems varied across the full range of the scale (1–5), indicating a heterogeneous participant pool.

Procedure. Participants interacted with the demo of the application and explored the generated environments. After completing the interaction, participants filled out a questionnaire consisting of:

- Likert-scale questions, where they rated 1–5 aspects such as the overall concept, visual quality, ease of understanding, and immersion.
- Multiple choice questions covering UI feature options (such as the menu type) or motion sickness
- Open-ended questions for additional feedback

Quantitative results. Overall, participants rated the general idea of the application very positively, with responses concentrated toward higher values (4–5). The visual quality of the generated worlds was likewise rated highly. Ratings for ease of understanding/usability were positive with slightly higher variance, suggesting that some users required additional time to become familiar with the interaction model. Finally, participants reported a generally high level of immersion during exploration. Figure 1 summarizes the distribution of Likert responses for these core dimensions.

Qualitative feedback. Qualitative responses highlighted several recurring themes. A subset of participants reported hesitation during exploration due to reduced awareness of their physical surroundings, concerns about real-world obstacles, or motion discomfort. Participants also suggested improvements such as clearer safety cues, reduced motion-related discomfort, and additional interaction/customization options.

Participants were also asked whether they would prefer a static menu or a menu tied to the hand. Out of 15 responses, 11 participants (73.3%) preferred a hand-tied menu, while 4 participants (26.7%) preferred a static menu.

Overall 81.3% of the respondents did not experience any form of Motion sickness, while 18.7% did experience it to some degree.

4.3. Rendering Performance

We evaluated the rendering performance of our system using a 3D Gaussian splat environment derived from *Interior in Vitre* by Nicolae Grigorescu.

We evaluated multiple versions of the scene with different Gaussian splat counts, ranging from 2.5 million to 250 thousand splats. The 500K-splat configuration corresponds to the low quality version generated directly by the Marble pipeline, while the remaining configurations were measured on scenes adapted by us. In order to generate the lower splat count scenes, we use the high quality 2.5 million splat file provided by Marble, and a simple python script to reduce the splat count. To maintain some structure within the scene, instead of just culling splats randomly, we sort the splat by a custom factor consisting of their opacity multiplied with their scale, ensuring that the largest and most opaque splats remain.

For the measurement, we implemented a live FPS display, displaying and storing the current FPS, as well as a running average, minimum, and maximum. Each scene was explored during a 2-minute window, allowing for meaningful FPS averages.

Results. Table 1 summarizes FPS metrics for each configuration. Overall, the results show a clear trend of increasing frame rate with decreasing splat count. High-complexity scenes (above 1 million splats) operate below real-time frame rates on average, while lower-complexity configurations achieve substantially higher performance, exceeding real-time thresholds. With lower splat counts, the scenes visual fidelity becomes progressively worse, with simple culling causing considerable holes to appear in the scene. The 500k-splat generated natively by Marble exhibits a significant decrease in performance compared to the culled 500k-splat, but retains a considerably higher visual

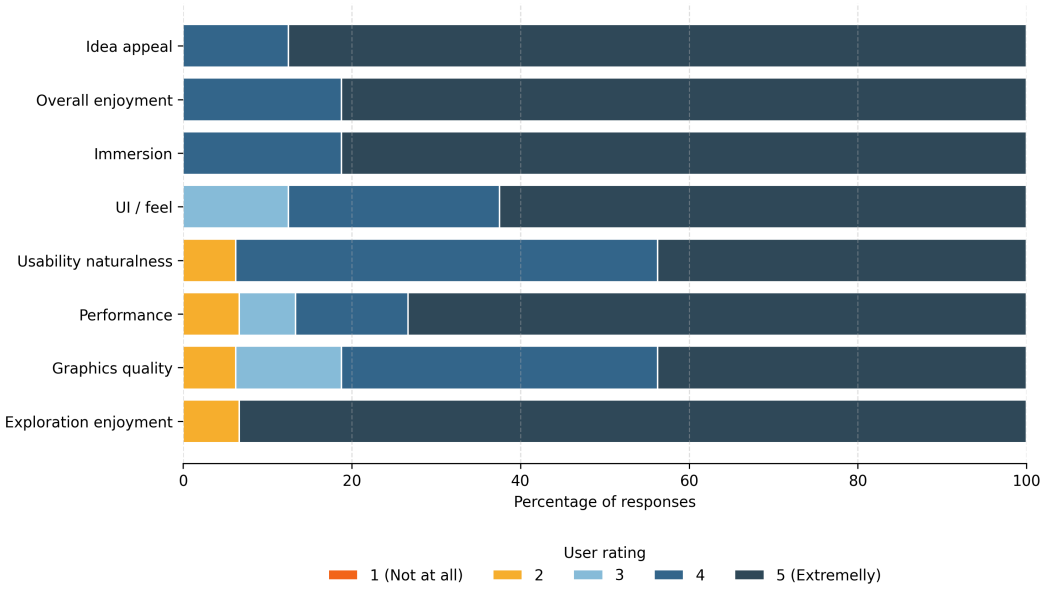

Figure 1. Distribution of responses (1–5) for the core demo-evaluation questions

quality, see figure 5. In fact, it maintains a higher quality even when compared to the 1.0M-splat, as seen in figure 6. As one can see, the culled splats leave ”holes” in the scene. In order to mitigate this we attempted to scale the splats up and see if the expected decrease in visual quality is a justifiable tradeoff for the increase in performance. Figure 7 provides a visual comparison of the scaled 750k culled splat, which is the scene with the highest splat count while providing a higher frame rate to the 500k Marble one. Even with aggressive scaling, the lack of splats cannot be easily mitigated, and the resulting scene looks drastically worse than the Marble version. The decrease in performance for the Marble 500k splat is most likely due to their more complex culling method and not increasing scales uniformly, which can also come with a performance drawback. After these tests, we ended up favoring the lower quality version provided by Marble over any of our culled ones, as it retained higher visual fidelity with a performance that was still acceptable.

5. Conclusion

In this paper, we presented V-ART, an augmented reality museum companion app that integrates Gaussian splat-based 3D worlds into a real museum-going experience using the Meta Quest 3. By combining painting metadata, spatial localization, QR-based interaction, and immersive, explorable environments, V-ART demonstrates how recent advances in generative 3D representations can be meaningfully embedded into cultural spaces rather than isolated VR experiences. Our implementation shows that, despite current limitations in tooling and performance, Gaussian splats can be rendered at usable quality on standalone VR hardware and integrated into a full-featured application. Results from our user study indicate strong user interest, high perceived immersion, and positive reception of the overall concept, while also highlighting minor areas for improvement related to safety, usability, and comfort. Overall, V-ART illustrates the great potential of AR companions to enrich museum visits and serves as a foundation for future work on interaction design, immersive experiences, and scalable deployment into real-world exhibitions.

Table 1. Rendering performance for different Gaussian splat counts using the *Interior in Vitre* scene.

Splats	Avg FPS	Min FPS	Max FPS
2.5M (Marble)	10.27	7.27	17.65
1.5M	13.20	9.90	20.26
1.0M	18.25	11.60	30.00
750K	22.64	16.63	35.26
500K	32.65	22.82	66.99
500K (Marble)	15.31	11.30	25.50
250K	52.14	33.20	123.97

References

[1] Nikolai Kalischek, Michael Oechsle, Fabian Manhardt, Philipp Henzler, Konrad Schindler, and Federico Tombari. Cubediff: Repurposing diffusion-based image models for panorama generation, 2025. 2

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42

(4), 2023. 1

[3] Meta Platforms, Inc. Meta mixed reality utility kit (mruk). https://developers.meta.com/horizon/documentation/unity/unity-mr-utility-kit-overview/, 2025. version used: 81.0.0. 3

[4] Microsoft Corporation. Mixed Reality Toolkit (MRTK). https://github.com/MixedRealityToolkit/MixedRealityToolkit-Unity, 2024. version used: 3.2.0. 3

[5] Aras Pranckevičius. Unitygaussiansplatting: Toy gaussian splatting visualization in unity. https://github.com/aras-p/UnityGaussianSplatting, 2025. GitHub repository, MIT License. 1

[6] Unity Technologies. Unity game engine. https://unity.com/, 2025. version used: 6000.0.62f1. 1

[7] Lee Alan Westover. *Splatting: a parallel, feed-forward volume rendering algorithm*. PhD thesis, University of North Carolina at Chapel Hill, 1991. 1

[8] World Labs (Fei-Fei Li et al.). Marble: A multimodal world model. World Labs Blog, 2025. 2

[9] Zhongqi Yang, Wenhang Ge, Yuqi Li, Jiaqi Chen, Haoyuan Li, Mengyin An, Fei Kang, Hua Xue, Baixin Xu, Yuyang Yin, Eric Li, Yang Liu, Yikai Wang, Hao-Xiang Guo, and Yahui Zhou. Matrix-3d: Omnidirectional explorable 3d world generation, 2025. 2

[10] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T. Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image, 2025. 2

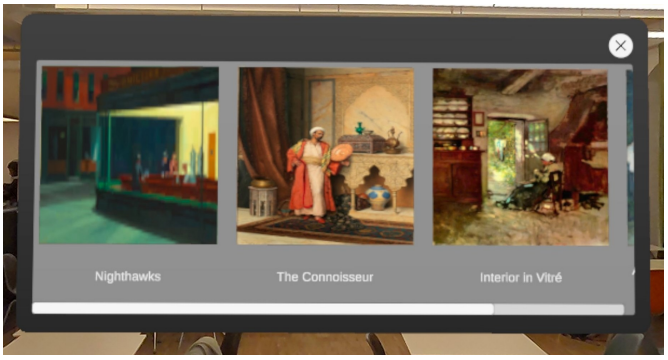

Figure 2. The painting collection user interface. This menu allows users to select an available painting to learn more about it.

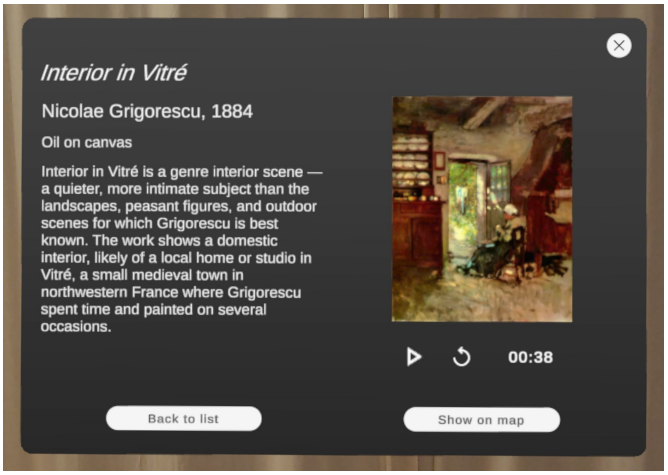

Figure 3. The painting information screen. Typical painting information is displayed and the user is given contextual buttons to view the paintings location or perform other tasks. Audio controls allow the user to play additional audio information.

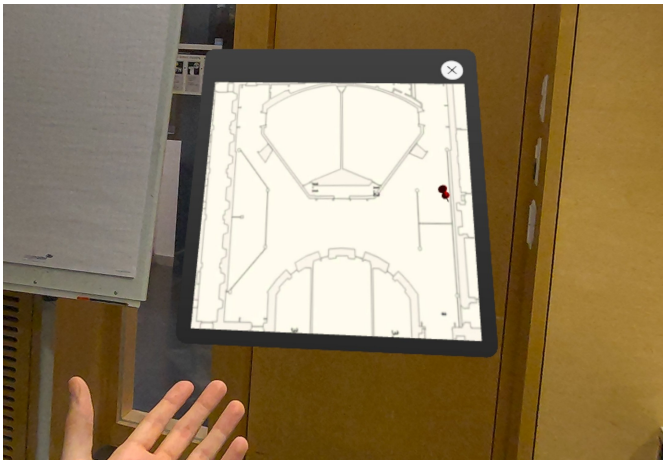

Figure 4. The in-app map. The map can display locations of paintings for easier navigation around the museum.

(a) Screenshot of a 500k Gaussian splat scene generated by Marble.

(b) Screenshot of a 500k Gaussian splat scene with a culled splat-count. The original scene contained 2.5 million splats.

Figure 5. A visual comparison of two 500k Gaussian splat scenes.

(a) Screenshot of a 500k Gaussian splat scene generated by Marble.

(b) Screenshot of a 1 million Gaussian splat scene with a culled splat-count. The original scene contained 2.5 million splats.

Figure 6. A visual comparison of two Gaussian splat scenes. Despite its far inferior visual quality, the 1 million splat scene showed comparable performance.

(a) Close-up screenshot of a 500k Gaussian splat scene generated by Marble.

(b) Close-up screenshot of a 750k splat with a scaling factor of 2. The original scene contained 2.5 million splats.

Figure 7. A visual comparison of two Gaussian splat scenes. The scaled version comes at a slight performance increase, but at the cost of drastically reduced visual quality.