ARTHUB: AN INTERACTIVE VISUAL ART GALLERY TO DISPLAY STUDENT WORKS AND SUPPORT ART EDUCATION USING VIRTUAL REALITY

Ada Zhou[1], Joshua Lai[2]

[1]Portola High School, 1001 Cadence, Irvine, CA 92618
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

The expression of art is incredibly important for young students. Equally important is allowing these students to showcase what they've done to peers and others. It helps them gain confidence in their own abilities and gives them motivation to practice their skills. The best way to do this is typically an in-person art gallery, but this is not always convenient. Those who wish to view the art may not always have the opportunity to travel to a physical location, and there are numerous costs associated with setup and cleanup. To address this issue, we propose ArtHub, a hybrid desktop and virtual reality application that allows anyone to upload their artwork to a virtual gallery that may then be explored by others. It makes use of the Unity game engine and is compatible with the Meta Quest 2 headset. In order to allow different people to view each other's work, it features a remote database that users may save their artwork to. Galleries are associated with a passcode to prevent different galleries from interfering. Unlike physical galleries ArtHub may be accessed from anywhere, requires little logistical setup, and is infinitely expandable. Our application will help support young artists in schools who otherwise may not have had the chance to show their work to a wider audience.

KEYWORDS

Art, Virtual Reality, Education, Gallery

1. INTRODUCTION

Art can capture human emotions and expressions that words are simply unable to. It is an incredibly important part of our society, and many experts argue that it is a defining factor of what makes humanity so unique from other species on this planet [3]. As such, we should put considerable amounts of effort into fostering the next generation of artists. After all, every great artist was young at some point. One of the best ways to support young students is to simply allow them to showcase their work. According to a study by researchers at Marmara University, students allowed to participate in art studies had significantly higher levels of self-confidence [1].

Oftentimes, the best way to experience a piece of art is by viewing it in person. However, there are some major problems that exist when it comes to conventional physical galleries. One issue is that it might not be convenient for people living far away to visit either an open house or an art gallery [6]. If a student's artwork is being presented in an open house, nearby friends and

David C. Wyld et al. (Eds): AdNLP, CDKP, NCO, SAI, MLT, Signal, SOFT, ICAITA, CMC - 2024
pp. 213-223, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141019

relatives must come in person to see their art. However, this is time-consuming and difficult, as there is a lot of preparation and cleanup that comes with creating an in-person art gallery experience; something many students and teachers do not have the time to do. And what if people still want to experience the art in an immersive manner, but are prohibited from traveling from place to place, as was the case in the 2020 Coronavirus pandemic [14]?

The Virtual Art Gallery allows users to upload custom artwork to a virtual 3D space, but unlike ArtSteps it does not support virtual reality headsets. This means that the level of immersion is greatly decreased and is not nearly as effective for fostering art appreciation as a VR environment. The Meta VR Gallery does support headsets and is quite immersive, but unlike our app it does not allow the user to change the content of the gallery. Images are fixed and thus the gallery serves only as a way to appreciate historical works of art but not the works of young students. Artsteps offers more features than the previous two, providing both headset support and allowing the upload of custom artwork, but custom uploads must be done by sending plans to the company and waiting for a response, which is time consuming and adds extra expenses. Our app, ArtHub, bypasses this middleman and allows direct uploads with no fuss or extra cost.

Our solution: ArtHub

To resolve the issues of logistics and physical distance presented above, we propose the ArtHub app. This application is a virtual 3d art gallery which aims to simulate the experience of an in-person museum. Users take on a first-person perspective, with freedom to move around the gallery as they please. The main advantage of the virtual gallery is that there is no need to travel a large distance to reach an on-site location, which is an issue for students without reliable access to transportation [15]. Instead it can be accessed from anywhere as long as an internet connection is available. For those wishing for a more immersive experience, a VR headset may be used to make it seem like the user is really there. This is better than a simple 2d interface, as studies have shown that art appreciation in students is greatly increased when viewed in a simulated 3d environment [5] and that it has a large impact on emotional perception of the objects in question being viewed [13]. However, for those who are unable to use a VR headset, either due to health or financial constraints, a conventional computer interface is also available. This can not only help those who cannot make it in person, but also those who have trouble appreciating physical art due to visual impairments [10]. Another draw of this virtual gallery is that it is much easier to set up than a real in-person gallery. There is no need to rent out a physical building, no need to clean up after untidy guests, and no need to protect the art pieces from potentially reckless viewers. A logistical nightmare suddenly becomes much more manageable for already busy students and teachers. Furthermore, its ease of access means that those setting up the gallery do not need to be nearly as selective with what art they choose to display. If setting up more pictures takes exponentially more effort, you must pick and choose what deserves the spotlight. The virtual world does not come with the same restrictions of space, structural stability, and need for human accommodation like the real world does, and as such it may focus entirely on the art itself. Our gallery is mainly targeted at students and teachers, giving schools the means to support their young aspiring artists.

Experiment A focused on testing network speed's impact on viewing and uploading artwork within the virtual art gallery app. By simulating different network conditions and measuring upload and download times for varying artwork file sizes, it revealed a relatively linear relationship between file size and latency, with inconsistencies increasing for larger files. This suggests that while network performance generally decreases with larger payloads, factors like network stability and data handling efficiency become more critical with increasing file sizes.

Experiment B aimed to assess the user interface's (UI) effectiveness in fostering user engagement and satisfaction. Through task-based user testing with 10 participants and analysis of engagement metrics, the experiment uncovered significant variability in task completion times and overall satisfaction ratings. The wide range in upload times and the high variability in user interactions indicated that the uploading process's efficiency and the intuitive design of the UI are crucial for enhancing user experience.

Both experiments highlighted the importance of optimizing backend processes and front-end designs to improve user satisfaction and engagement within the app, suggesting areas for targeted improvements.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. How to Store the Art

One problem that would be faced in the development of this application would be how to store the art and make sure other people can see it. In order to solve this problem, we could use traditional multiplayer game methods and have one device act as a “server” while other devices connect to it. The data could then be streamed to the client devices [11]. Alternatively, we could use a remote web database. This means that we would not be able to implement live interaction between players, but it would be more stable and less prone to data loss [12].

2.2. How to Avoid Conflicts between Different Galleries

Another problem that might arise is how to maintain order and avoid conflict when there are many concurrent users and galleries, each with different purposes. To solve this issue, we could allow users to decide how they would like to save their gallery, either storing it as a local file on their own device if they simply want to create a gallery for personal uses, or a cloud upload should they decide to share it with others. We could create a password system to allow users to save different galleries without worrying about interference from other users.

2.3. How to Visually Display the Art

Finally, there is the issue of actually displaying the art to the player in an interesting and immersive way. One way to solve this problem is to use a game engine to simulate a 3D environment and build a visual gallery which can display artwork. This gallery could be made accessible through the use of a virtual reality headset in order to further improve immersion. Additionally, to improve interactivity, we could add features such as zooming into the picture, viewing description of the art pieces, etc.

3. SOLUTION

My gallery is built using the Unity 3D game engine and is programmed in C#. When the user opens the Art Gallery, they are presented with a menu that contains several options. From there, they may choose to enter a password to define which gallery they want to access. When they press the “load” button, a request is sent to a Firebase database, and it fetches the data associated with the corresponding password. Once the user can see pictures placed in the frames around the gallery, they may walk around and view them, clicking on them to zoom in and get a better look. If they wish to add pictures, they may select the option from the menu, then upload from either

their own computer or by using an internet URL. Next, they can define a title, artist name, and description for the picture. After all of this information is filled out, they can walk around the gallery and click on a picture frame to place it in. Additionally, users may delete pictures if they wish. From there, they may save the gallery, either to their local device, or to the remote database as they please. If saving locally, the system will compress the data into a readable format and save it as a single file in a directory located on the user's device. If saving to the cloud, the system instead passes this save package to a Firebase communicator script, which sends requests one by one to the database containing the picture information.

Figure 1. Overview of the solution

Picture Frame

The picture frames are the most important part of the gallery, as their purpose is to act as a modular component which holds the artwork itself. We designed four different variations of picture frames to accommodate different picture sizes including large, small, horizontal and vertical. They contain interactive elements in the form of buttons to allow the user to perform various actions on them, such as zooming in, placing pictures inside, etc.

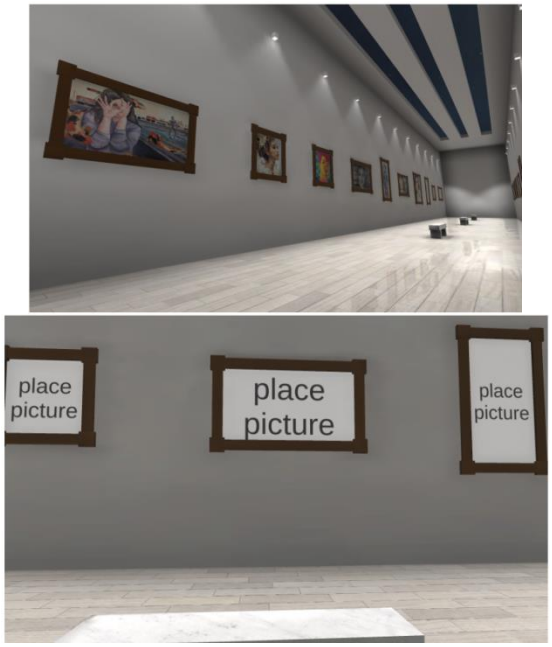

Figure 2. Place picture

```
public void PlacePicture()
{
    canvas.material.SetTexture("_MainTex", ImageManager.Main.LoadedTexture);
    ImageManager.Main.ImagePlace();
    nameText.text = ImageManager.Main.ArtistName;
    print(ImageManager.Main.ArtistName.ToString());
    descriptionText.text = ImageManager.Main.description;
    titleText.text = ImageManager.Main.title;
    HasImage = true;
}

public void RemovePicture()
{
    nameText.text = "";
    descriptionText.text = "";
    titleText.text = "";
    canvas.material.SetTexture("_MainTex", defaultTexture);
    ImageManager.Main.ImageRemove();
    HasImage = false;
}
```

```
//take all your stuff and put it on a piece of paper
public PictureInformation GetData()
{
    PictureInformation NewInformation = new PictureInformation();
    NewInformation.description = descriptionText.text;
    NewInformation.ArtistName = nameText.text;
    NewInformation.title = titleText.text;
    NewInformation.texture = ImageConversion.EncodeToJPG((Texture2D)canvas.material.GetTexture("_MainTex"));
    NewInformation.id = id;
    return NewInformation;
}
//take a piece of paper, read it and put all of the information on you
public void SetData(PictureInformation information)
{
    descriptionText.text = information.description;
    nameText.text = information.ArtistName;
    titleText.text = information.title;
    Texture2D texture = new Texture2D(1, 1);
    if(information.texture == null)
    {
        texture = defaultTexture;
        HasImage = false;
    }
    else
    {
        texture.LoadImage(information.texture);
        HasImage = true;
    }
    canvas.material.SetTexture("_MainTex", texture);
}
```

Figure 3. Screenshot of code 1

This code is part of the PictureController script, which is placed on every picture frame. The methods seen above manipulate the data shown in the picture frame and may be accessed externally from a different script. The PlacePicture() method reads in the picture that the player selected earlier through the Image Manager, then renders it in the frame. It also reads the name, description, and title set by the player and puts them in as well. The RemovePicture() method, as the name suggests, clears out all data from the picture frame. The GetData() and SetData() methods are used by the Image Manager to both collect data from the frames in order to save them, or to give each frame a data package to tell it what to render in the frame. In the GetData() method, a PictureInformation class is created and all of its fields are filled with the relevant data. The image is encoded into a JPG file to save on space. The SetData() method will load the default texture if no texture is provided.

The Image Manager is the central script of the Art Gallery. It manages all of the picture frames and acts as an interface between the frames and the Firebase Manager. When a user wishes to place an image in the gallery, the Image Manager will handle loading the texture or downloading from the web if the user puts a link. It also houses the controls for inputting the name, title, and description of the picture to be placed.

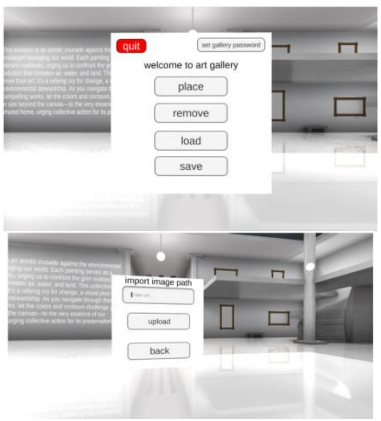

Figure 4. Screenshot of main pages

```
public void SubmitInfo()
{
    description = DescriptionInput.text;
    ArtistName = ArtistNameInput.text;
    title = TitleInput.text;
    OnImageLoad.Invoke();
}
```

```
SavePackage GenerateSavePackage()
{
    List<PictureController> PictureList = new List<PictureController>();
    PictureController[] AllPictures = FindObjectsOfType<PictureController>();
    foreach (PictureController pic in AllPictures)
    {
        if (pic.HasImage)
        {
            PictureList.Add(pic);
        }
    }
    PictureController[] pictureControllers = PictureList.ToArray();
    PictureInformation[] StackOfPaper = new PictureInformation[pictureControllers.Length];
    for (int i = 0; i < pictureControllers.Length; i++)
    {
        StackOfPaper[i] = pictureControllers[i].GetData();
    }
    SavePackage DataToSave = new SavePackage();
    DataToSave.StackOfPaper = StackOfPaper;
    DataToSave.score = 10;
    return DataToSave;
}
```

```
public void LoadDataLocal()
{
    if (File.Exists(Application.persistentDataPath + "/savedata.dat"))
    {
        PictureController[] pictureControllers = FindObjectsOfType<PictureController>();
        foreach (PictureController pc in pictureControllers)
        {
            pc.RemovePicture();
        }
        print("load in data");
        FileStream file = File.Open(Application.persistentDataPath + "/savedata.dat", FileMode.Open);
        BinaryFormatter bf = new BinaryFormatter();
        SavePackage loadeddata = (SavePackage)bf.Deserialize(file);
        PictureInformation[] loadedstack = loadeddata.StackOfPaper;
        foreach(PictureInformation pieceofpaper in loadedstack)
        {
            foreach(PictureController pc in pictureControllers)
            {
                if(pc.id == pieceofpaper.id)
                {
                    pc.SetData(pieceofpaper);
                }
            }
        }
    }
    else
    {
        print("no save data");
    }
}
```

Figure 5. Screenshot of code 2

The SubmitInfo() method is called whenever the user finishes entering all their data (name, title, description) for a certain picture. The method records all the data, then sends out an event notifying all of the picture controllers that they should get ready to receive a picture. The GenerateSavePackage() method is used when the user wishes to save their gallery. It takes all of the pictures and text in each picture frame, assigns them IDs based on their position in the world, and compresses it into a serializable format that can then be saved either locally as a file or in our remote Firebase database. The LoadDataLocal() method is what is used when the SavePackage is to be retrieved from a place on the user's physical drive. It first searches for the save package, and if it exists, it clears out the gallery, reads the package into a SavePackage class, then assigns each of the picture frames' data by matching up their ID's.

The Firebase communicator handles all interactions over the web with our Firebase Database. It manages both uploading data to the database and downloading from it. The communicator interfaces with the Image Manager to perform its functions by sending the manager SavePackages which the communicator receives, and by receiving SavePackages from the manager to send to the database.

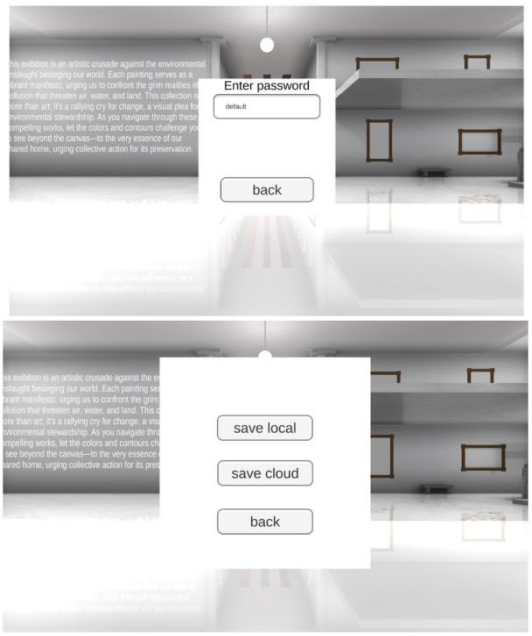

Figure 6. Screenshot of save page

```
public void UploadData(SavePackage savePackage)
{
    PictureInformation[] PaperStack = savePackage.StackOfPaper;
    for (int i = 0; i < PaperStack.Length; i++)
    {
        PictureInformation paper = PaperStack[i];
        DatabaseReference PaperEntry = DataBaseRoot.Child(password).Child("pictures").Child(paper.id);
        PaperEntry.Child("ArtistName").SetValueAsync(paper.ArtistName);
        PaperEntry.Child("ArtName").SetValueAsync(paper.title);
        PaperEntry.Child("Description").SetValueAsync(paper.description);
        string PictureString = "";
        if (paper.texture != null)
        {
            PictureString = Convert.ToBase64String(paper.texture);
        }
        PaperEntry.Child("Texture").SetValueAsync(PictureString);
    }
}
```

```
public void DownloadData()
{
    Debug.Log("loaded");
    SavePackage package = new SavePackage();
    DataBaseRoot.Child(password).Child("pictures").GetValueAsync().ContinueWith(task =>
    {
        if (task.IsCanceled)
        {
            Debug.Log(task.Exception);
            return;
        }
        if (task.IsFaulted)
        {
            Debug.Log(task.Exception);
            return;
        }
        DataSnapshot dataSnapshot = task.Result;
        package.StackOfPaper = new PictureInformation[dataSnapshot.ChildrenCount];
        int index = 0;
        foreach (DataSnapshot snap in dataSnapshot.Children)
        {
            PictureInformation paper = new PictureInformation();
            paper.id = snap.Key;
            paper.ArtistName = snap.Child("ArtistName").Value.ToString();
            paper.title = snap.Child("ArtName").Value.ToString();
            paper.description = snap.Child("Description").Value.ToString();
            string texturestring = snap.Child("Texture").Value.ToString();
            if (texturestring == "")
            {
                paper.texture = null;
            }
            else
            {
                paper.texture = Convert.FromBase64String(texturestring);
            }
            package.StackOfPaper[index] = paper;
            index += 1;
        }
        ImageManager.Main.FirebaseReciever(package);
    }, TaskScheduler.FromCurrentSynchronizationContext());
}
```

Figure 7. Screenshot of code 3

These two-methods control loading and downloading data from/to the database. The UploadData() method receives a SavePackage from the ImageManager, and first begins by deconstructing the package into its individual PictureInformation classes. Then, it locates the directory in which the data should be saved in the database. Next, for each PictureInformation class, it asynchronously sets the corresponding field in the database with the appropriate data. The DownloadData()

method does the same thing, but in reverse. It first checks to see if the save data directory in the database is valid, then creates an empty SavePackage. It then asynchronously reads from the database, filling out the SavePackage with the appropriate data. Since the operation is asynchronous, it must call a method once it finishes to indicate that it is done. It does this by calling the FirebaseReceiver() method in the ImageManager, which will take things from there.

4. EXPERIMENT

4.1. Experiment 1

Experiment A is designed to evaluate the impact of network speed on user experience in the virtual art gallery, specifically focusing on the latency involved in viewing and uploading artwork. The experiment A aims to assess network performance impact on the virtual art gallery's user experience, focusing on latency during artwork viewing and uploading. We'll use a diverse set of digital artworks with varying file sizes to simulate real-world usage. An automated script will upload and download these artworks under different network conditions (high-speed, 4G, slow Wi-Fi) to Firebase, logging response times. The analysis will compare these times against a control dataset obtained under optimal conditions to identify performance bottlenecks. This approach will inform necessary optimizations for improving user engagement and satisfaction across various network speeds.

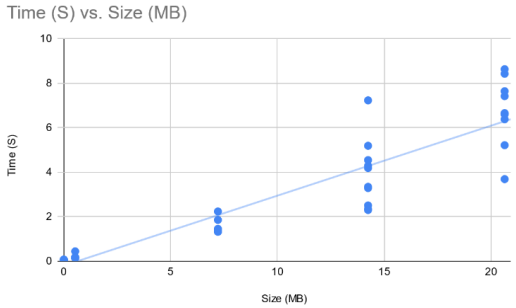

Figure 8. Figure of experiment 1

Statistics for each trial:

0MB: 0.069
0.53MB: 0.196
7.21MB: 1.530
14.24MB: 3.927
20.62MB: 6.741

In analyzing the network performance data for the virtual art gallery, we observed that as the file size of uploaded artwork increases, so does the latency in upload times, demonstrating a relatively linear relationship between payload size and response time. Here's a breakdown of the key findings:
Mean and Median Upload Times: The data indicates increasing upload times with larger file sizes, suggesting a direct correlation between file size and network latency. The mean and median values progressively rise, reflecting this trend.

Range of Values: The quickest upload time was recorded at 0.069 seconds for a 0MB file (essentially an empty file), indicating minimal processing time. The longest upload time was 6.741 seconds for a 20.62MB file, showcasing the impact of file size on upload duration.

Observations on Consistency: Initial tests with smaller files (up to 0.53MB) showed consistent upload times. However, as file sizes increased beyond 7.21MB, the variability in upload times also increased, suggesting that larger files introduce more unpredictability in network performance.

Surprising Data: The relationship between file size and upload time was expectedly linear for smaller files but showed signs of increasing inconsistency with larger sizes. This variance was unexpected and points to potential inefficiencies in how large data payloads are managed.
Influence on Results: The most significant factor affecting upload times was the file size of the artwork. However, the increasing inconsistency with larger files suggests other influencing factors at play, possibly including network stability and the efficiency of data handling mechanisms within the gallery app and Firebase.

These findings highlight the importance of optimizing data transfer protocols for large files and investigating further into the backend processes to ensure consistent performance across all user interactions within the virtual art gallery.

4.2. Experiment 2

Experiment B is designed to evaluate the effectiveness of the user interface (UI) design in promoting user engagement and interaction within the virtual art gallery app.

Experiment B aims to assess the virtual art gallery app's user interface (UI) design and its impact on user engagement. We will recruit 10 participants with varying art gallery experiences to perform specific tasks within the app, such as finding and uploading artwork. Engagement will be measured using analytics for task completion time and interaction frequency, supplemented by participant feedback on usability and design appeal. The goal is to identify UI elements that enhance or hinder user experience, providing insights for targeted improvements. This approach will highlight the UI's role in promoting user interaction and satisfaction within the virtual gallery environment.

Participant ID	Time to Find Artwork (s)	Time to Upload Artwork (s)	Interactions per Visit	Overall Satisfaction (1-5)
1	45	90	15	4
2	30	120	20	5
3	55	85	12	3
4	50	105	18	4
5	35	95	22	5
6	40	100	14	4
7	60	80	11	3
8	65	75	10	2
9	25	130	25	5
10	70	115	8	3

Figure 9. Figure of experiment 2

The analysis reveals mean times of 47.5 seconds to find artwork and 99.5 seconds for artwork upload, with median values closely mirroring these at 47.5 and 97.5 seconds, respectively. The lowest recorded times were 25 seconds to find and 75 seconds to upload artwork, with the highest at 70 and 130 seconds, respectively. Interaction rates varied from 8 to 25 per visit, with overall satisfaction ranging from 2 to 5.

Surprisingly, the broad range in upload times and the high variability in user interactions suggest significant differences in user experiences, possibly due to varying network conditions or differing levels of familiarity with the app. The most significant impact on the results appears to be the efficiency of the uploading process, as indicated by the wide range of upload times. Optimizing this aspect could significantly enhance user satisfaction and engagement, suggesting a need for further investigation into optimizing file handling and network efficiency.

5. RELATED WORK

The Virtual Art Gallery takes a similar approach to our app, allowing users to view artwork in a simulated space. Artists may upload their own custom work through an online portal, after which it is displayed on a website that other people may view [7]. One large drawback of this virtual art gallery is that it lacks compatibility with a VR headset. Users must view the gallery on a conventional flat-screened device, which is better than a 2d website, but cannot afford nearly the same level of immersion as being there in person and thus leads to decreased appreciation [4]. In comparison, not only does our art gallery run on a standard computing device, it can also run through the Meta Quest headset, allowing for a much more realistic and convincing virtual reality experience.

The Meta VR Art Gallery is on the Meta store, and as such can only be viewed through a Meta Quest headset. It is graphically realistic and allows the user to view a variety of paintings from famous historical artists such as Pierre-Auguste Renoir, Claude Monet and Rembrandt [8]. While it is well built with a considerable amount of care put into the layout, it is completely static and uncustomizable. All the galleries featured in this app cannot be changed or rearranged, and users may not upload their own custom art. In comparison, our application allows the user to upload any art they wish, and even allows them to save them online for others to view and also make their own contributions to.

Similar in concept to ArtHub, the Artsteps gallery allows users to walk around a 3d art gallery either in a headset or using a traditional computer interface. In order to get a gallery in Artsteps, a client must pick a room template, then send a floor plan and pictures to the company, after which they will design the gallery for the client [9]. Not only is this slow, since the client needs to wait for a response after sending their plan, but it is also expensive as each virtual gallery is hand-made by designers at the company. Some templates on their site were listed at over $200, way out of budget for most teachers. In comparison, our gallery allows the end-user to add/remove pictures themselves, with no need to wait or pay for a middleman.

6. CONCLUSIONS

Despite the positives, there are still some limitations in ArtHub. One of the major problems is that when uploading artwork, users must choose a specific frame size that matches their art's dimensions. Although we include frames for pictures in different sizes, the size might still not be completely accurate. Some artworks may be stretched to fill the picture frame. In order to fix this, we can either create a picture frame that automatically resizes itself to fit the content, or use a UI canvas to represent the picture to allow for automatic resizing. Another limitation is there is currently only one scene in ArtHub. If we want to promote our project to all kinds of users, we should establish various scenes for different topics. We may also need to rework our picture saving format to this, as right now the picture IDs are based on position so they would not be compatible between separate styles. Finally, the app's framerate may stutter from time to time, so there are definitely rendering optimizations that could be done.

Despite its limitations, this app offers a promising augmentation to physical art galleries. It solves numerous problems involving accessibility, price, and ease of setup. Overall, ArtHub can help give more students a voice and the confidence they need in this increasingly competitive world of art, fostering the next generation of great artists.

REFERENCES

[1] Nalbur, Vicdan. "Interdisciplinary art education and primary teaching students' self-confidence." Kı brıslı EğitimBilimleriDergisi 16.4 (2021): 2010-2024.

[2] Tranquilli, Paulette. "Improving Student Attitudes through Increased Confidence and Accountability in Art." (1999).

[3] Hirst, Paul H. "Literature and the fine arts as a unique form of knowledge." Cambridge Journal of Education 3.3 (1973): 118-132.

[4] Liu, Pei. "Application and Teaching Exploration of Virtual Reality Technology in Art Appreciation." International Journal of Learning and Teaching 3.7 (2021).

[5] Liu, Pei, and SathaPhongsatha. "Application research on enhancing the cognitive ability of art appreciation of senior high school students in Chengdu through virtual reality technology." International Journal of Innovative Research and Scientific Studies 5.3 (2022): 236-248.

[6] Allen, Jeff, and Steven Farber. "How time-use and transportation barriers limit on-campus participation of university students." Travel behaviour and society 13 (2018): 174-182.

[7] Chia, Ivy, and June Tay. "Virtual Art Gallery: A Multi-disciplinary Approach." International Journal of Multidisciplinary Perspectives in Higher Education 7.1 (2022): 98-113.

[8] Maddy, Joshua, and Husnu S. Narman. "Virtual Reality Museum Application for the Arts." 2023 IEEE Integrated STEM Education Conference (ISEC). IEEE, 2023.

[9] Chia, Ivy, and June Tay. "Virtual Art Gallery: A Multi-disciplinary Approach." International Journal of Multidisciplinary Perspectives in Higher Education 7.1 (2022): 98-113.

[10] Kwon, Nahyun, Yunjung Lee, and Uran Oh. "Supporting a crowd-powered accessible online art gallery for people with visual impairments: a feasibility study." Universal Access in the Information Society 21.4 (2022): 967-982.

[11] Ahde, Jani. "Real-time Unity Multiplayer Server Implementation." (2017).

[12] Chougale, Pankaj, et al. "Firebase-overview and usage." International Research Journal of Modernization in Engineering Technology and Science 3.12 (2021): 1178-1183.

[13] IJsselsteijn, Wijnand, et al., eds. Persuasive Technology: First International Conference on Persuasive Technology for Human Well-Being, PERSUASIVE 2006, Eindhoven, The Netherlands, May 18-19, 2006, Proceedings. Vol. 3962. Springer, 2006.

[14] Kumar, Vijay, et al. "Evaluating the impact of covid-19 on society, environment, economy, and education." Sustainability 13.24 (2021): 13642.

[15] Valant, Jon, and Jane Arnold Lincove. "Transportation inequities and school choice: How car, public transit, and school bus access affect families' options." Educational Researcher 52.9 (2023): 535-543.