Project Report

Ludwig Palette: An AR-enabled painting game to be played in Ludwig church

Hossein Monjezi
Prakash Naikade

Introduction and Motivation

Augmented reality (AR) is the experience of virtual elements in the real-world environment. With the help of AR technologies, changes can be applied to the objects, they can become interactive, and information can be displayed next to the artifacts. These interwoven virtual and real elements lead to an immersive and enhanced perception of the situation. Such immersion is one of the best options to improve the experience of visitors of historic places and museums.

Since our objective is to help visitors of the Ludwig church, a historical church in Saarbrücken, explore the environment of the church in an entertaining way, we have created an AR-enabled painting app to be played inside the church. Players of this game need to walk to the different areas of the church while painting the surfaces. This project is implemented in a team of two students, as a part of our Media Informatics curriculum. The functionality of the app is divided into two main parts, namely interaction and user interface. Each team member was responsible for one of the two parts. Additionally, we used issue-tracking, and automated Kanban project board features of Github to track our progress, as well as labels to add more information about the status, priority, and type of the issues. Although the workflow was suitable for the simultaneous development of the application, we did not work in parallel to avoid git conflicts.

Concept

The main idea of this app is to keep people engaged, help them explore the different areas of the church while paying attention to their surroundings. Moreover, we want to avoid hurried movements and loud noises. Painting serves this purpose perfectly. Although there are many possibilities to improve this idea, we have only created the minimum viable product (MVP). Currently, users can paint the surfaces, choose from a variety of colors, adjust the brush size, and undo and redo their actions. Also, their progress is saved automatically for later visits. Our vision for the final version of the app is to provide ways to share the results of painting on social media in the form of screenshots, and on the application itself in a way that visitors can also apply the paintings created by others to the church environment. The existing functionality will also be extended to include a smart color palette, and different brushes and stamps.

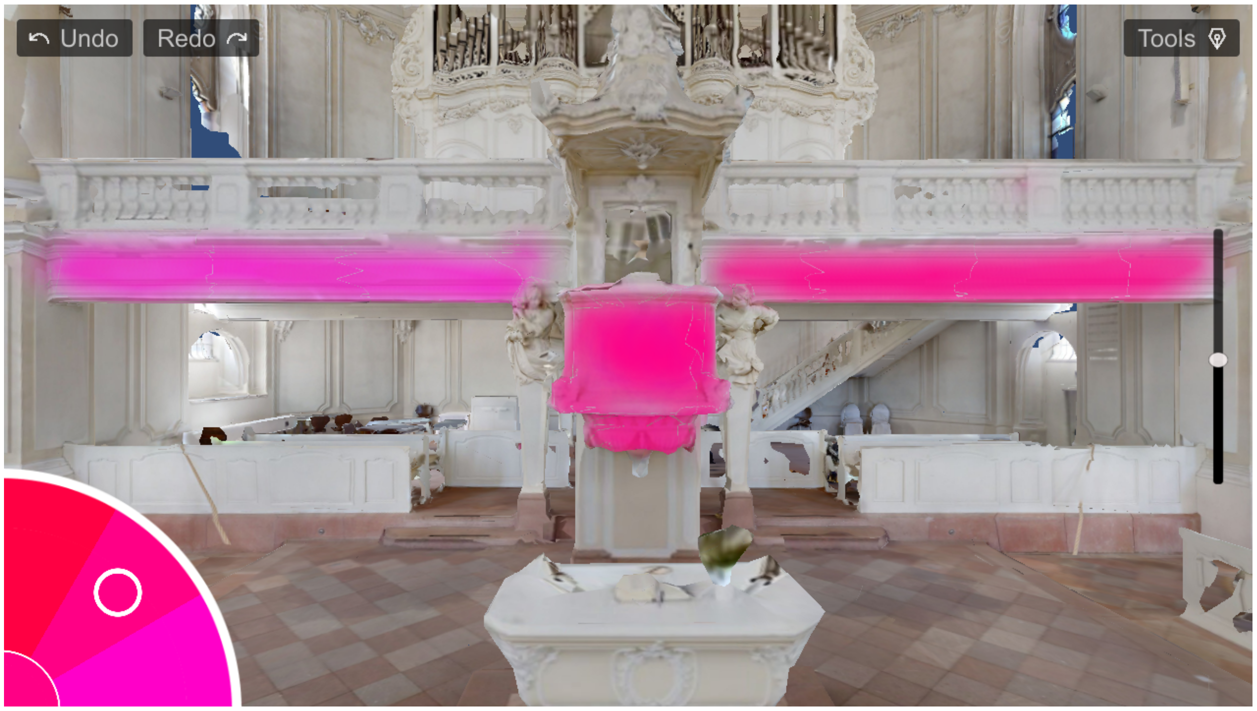
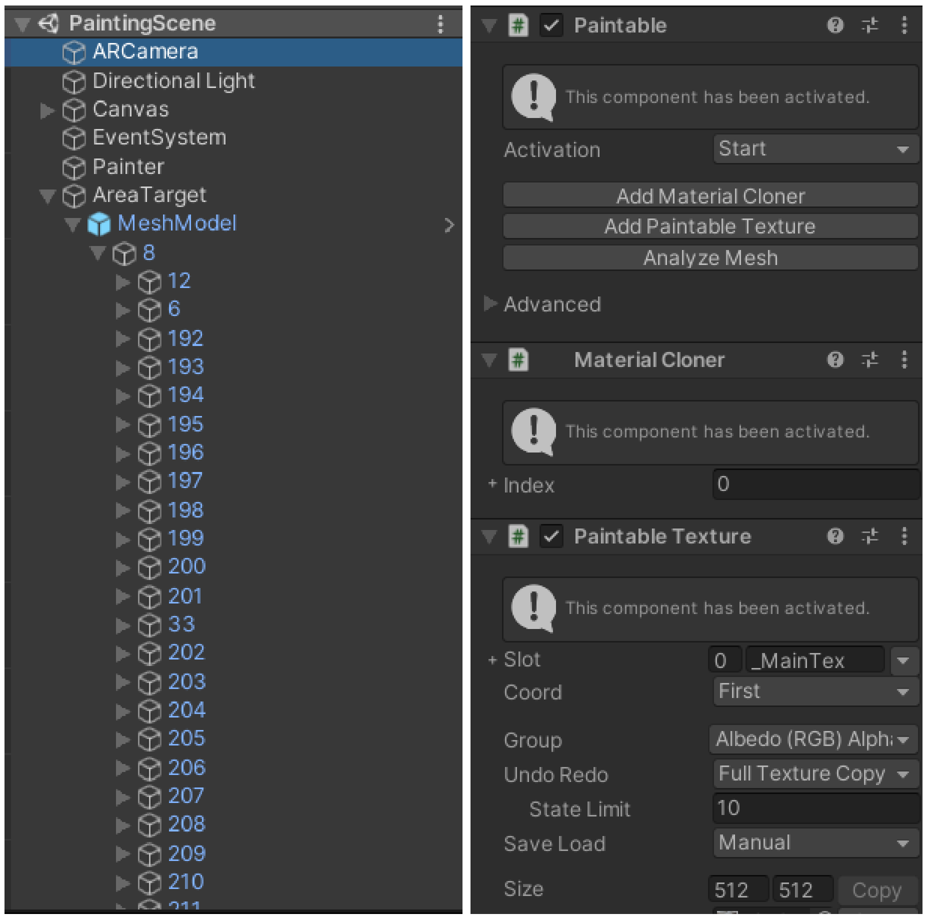


Figure 1

Implementation

To implement this project, we have used Unity Engine as our game engine, and Vuforia as the AR framework. Vuforia offers several features, including Area Target, which we used to detect the exact location of the player inside the church. To create an Area Target, you need a 3D scan of the target area, in our case, a scan of Ludwig church, and the area target generator application of Vuforia. This 3D model is used to create an Area target, which will be used by Vuforia to detect that specific environment, and to detect the exact location of the user. We have also used two plugins, namely Paint in 3D, and Another Color Picker.

Painting functionality was completely implemented using the Pain in 3D plugin. To make the area target paintable, we had to activate the mesh collider feature of the area target, then select all the meshes in the simulated mesh , and add the Paintable component from Paint in 3D to each of them. This component then adds MeshCloner, and PaintableTexture components (Figure 2), both of which will be used by the plugin itself. In addition to the paintable meshes, we needed to create a Painter gameobject as our brush. This Painter object uses the HitScreen component to define how the painting works (e.g. using ray casting), and the PaintSphere component to define the characteristics of the brush, such as color, and the size. Another Color Picker changes the color attribute in the PaintSphere component. In addition to the plugins, we also created BrushSize, and ShowHideTools components. BrushSize component is added to a slider, and is used to change the radius of the brush in the PaintSphere component. Since the color picker and the slider take a noticeable part of the screen space, we need a way to hide these tools, hence the ShowHideTools script is added to a button. As its name suggests, this component is responsible for toggling the visibility of the tools (Figure 3).



```
PaintingScene
  ARCamera
  Directional Light
  Canvas
  EventSystem
  Painter
  AreaTarget
    MeshModel
      8
        12
        6
        192
        193
        194
        195
        196
        197
        198
        199
        200
        201
        33
        202
        203
        204
        205
        206
        207
        208
        209
        210
```

```
Paintable
  This component has been activated.
  Activation            Start
  Add Material Cloner
  Add Paintable Texture
  Analyze Mesh
  Advanced

Material Cloner
  This component has been activated.
  + Index               0

Paintable Texture
  This component has been activated.
  + Slot                0  _MainTex
  Coord                 First
  Group                 Albedo (RGB) Alph
  Undo Redo             Full Texture Copy
    State Limit         10
  Save Load             Manual
  Size                  512  512  Copy
```

Figure 2

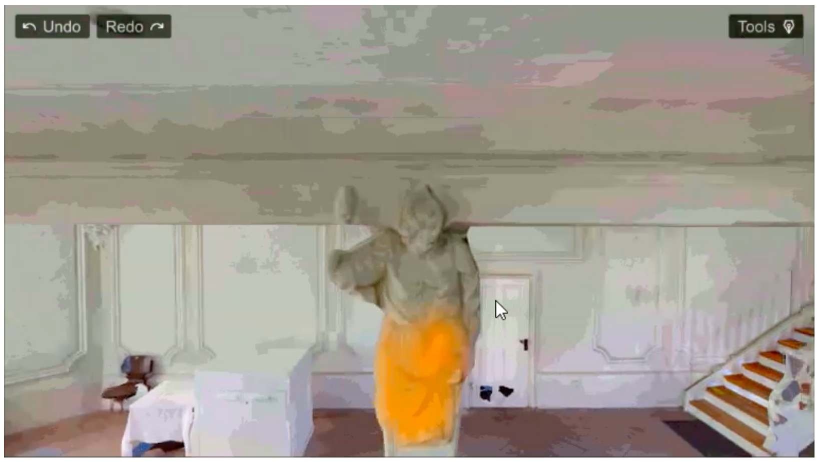
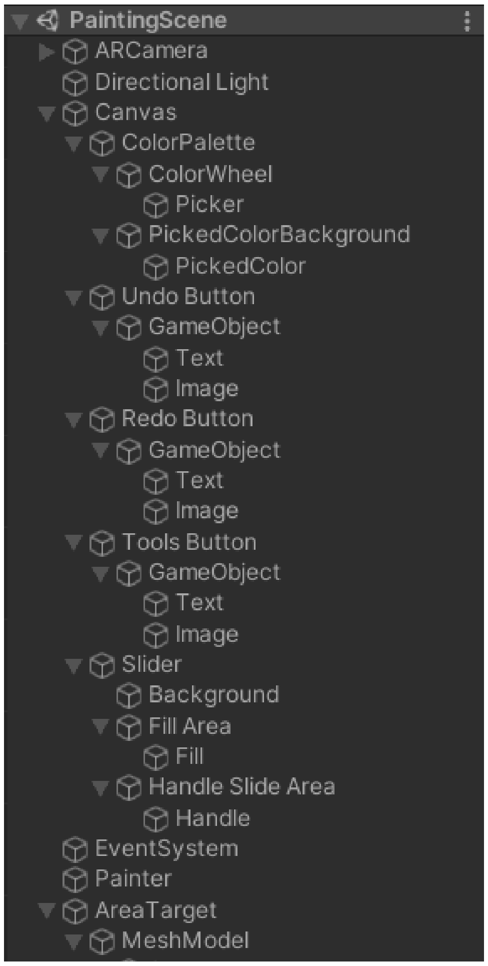


Figure 3

To make the user interface more visually appealing, we added a text and an icon to each button, which shows the functionality of the button (Figure 4). Moreover, the colors of the buttons and slider are changed to black, since it is the opposite of the dominant white color of the church. The feedback for the normal, pressed, and selected states of the buttons are visible through the change of their opacity.



```
PaintingScene
  ARCamera
  Directional Light
  Canvas
    ColorPalette
      ColorWheel
        Picker
      PickedColorBackground
        PickedColor
    Undo Button
      GameObject
        Text
        Image
    Redo Button
      GameObject
        Text
        Image
    Tools Button
      GameObject
        Text
        Image
    Slider
      Background
      Fill Area
        Fill
      Handle Slide Area
        Handle
  EventSystem
  Painter
  AreaTarget
    MeshModel
```

Figure 4

Lessons learned

We faced many challenges while implementing this project. The first challenge was the resource constraints of mobile devices, due to the number of meshes in the church area target. These meshes were considerably heavy, yet they were required to make the model paintable. Meanwhile, keeping all of them in the project would lead to its crash. We solved this problem by removing most of the meshes. Consequently, it was not possible to make the whole church paintable. The second problem was due to our choice of painting plugin, since it changes the texture of the objects. Even though it works perfectly in the simulator, there are no rendered objects in the church, hence the painting functionality will not work in AR. Although the result of our work (Figure 5 and Figure 6) could be exported as a VR game, we did not have any target devices, therefore, we could not choose an SDK to use or test our app.

Although the result was not what we hoped for, these challenges led to the realization that this idea is doable on a smaller scale. With our current knowledge, we know we should have used the model target of some interesting sculptures in the church, rather than the whole environment, to avoid issues with memory constraints. Also, we should have implemented our own painting feature, rather than using a plugin.

In conclusion, we were able to create a paintable church environment that could be used as a VR game, but there are some major changes required to make this game work in AR. These changes include creating a new painting feature, and scaling down the area to only sculptures, to avoid crashes due to the memory constraints.

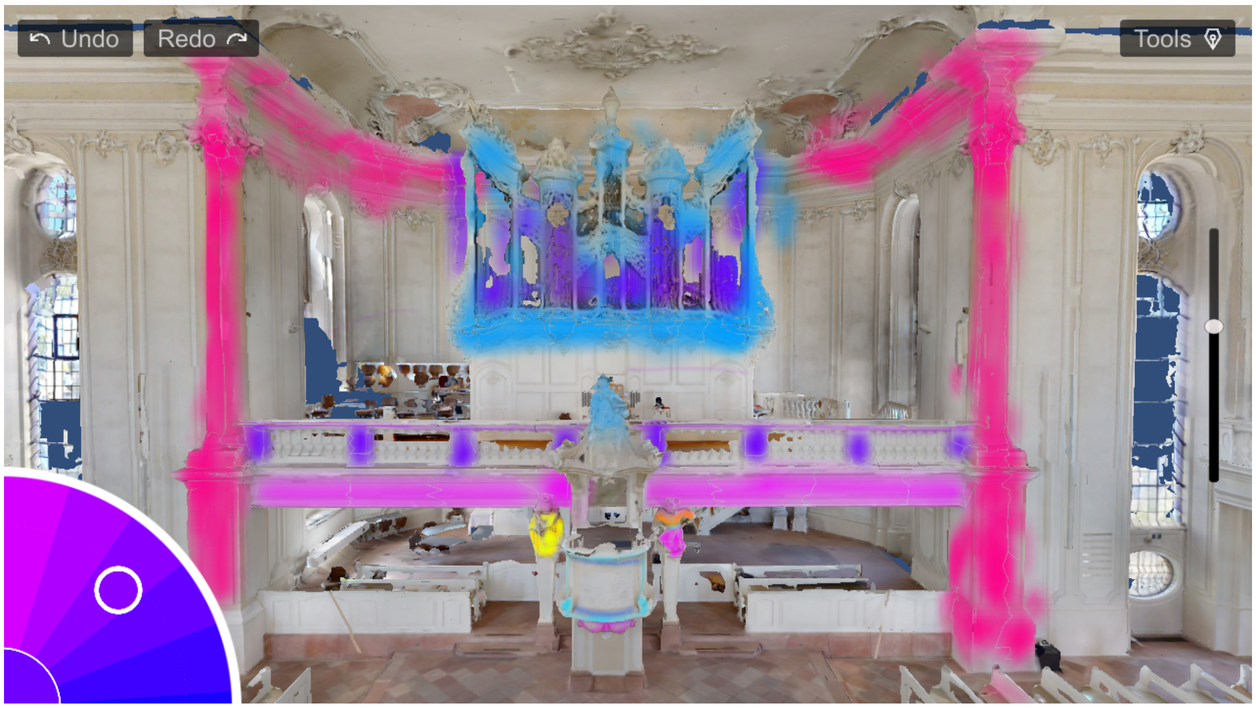


Figure 5

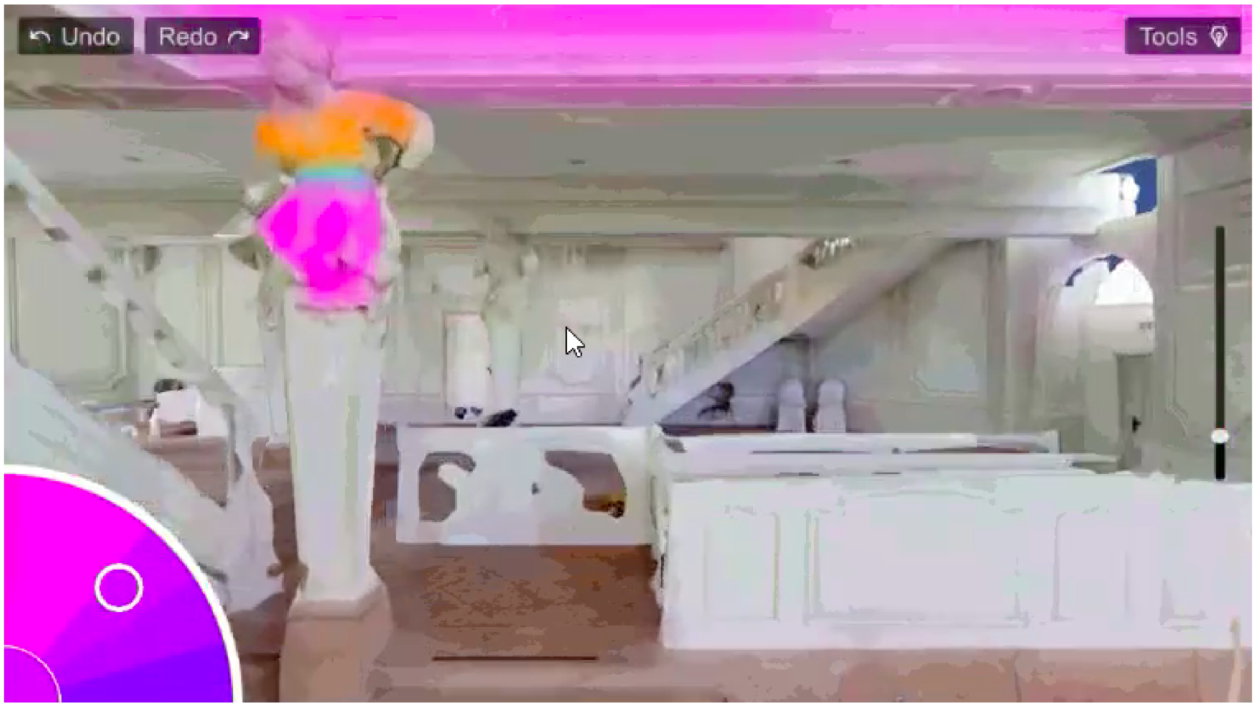


Figure 6