CHAR - Cultural Heritage AR

Dominik Huber, Felix Stieglbauer, David Plecher, Gudrun Klinker

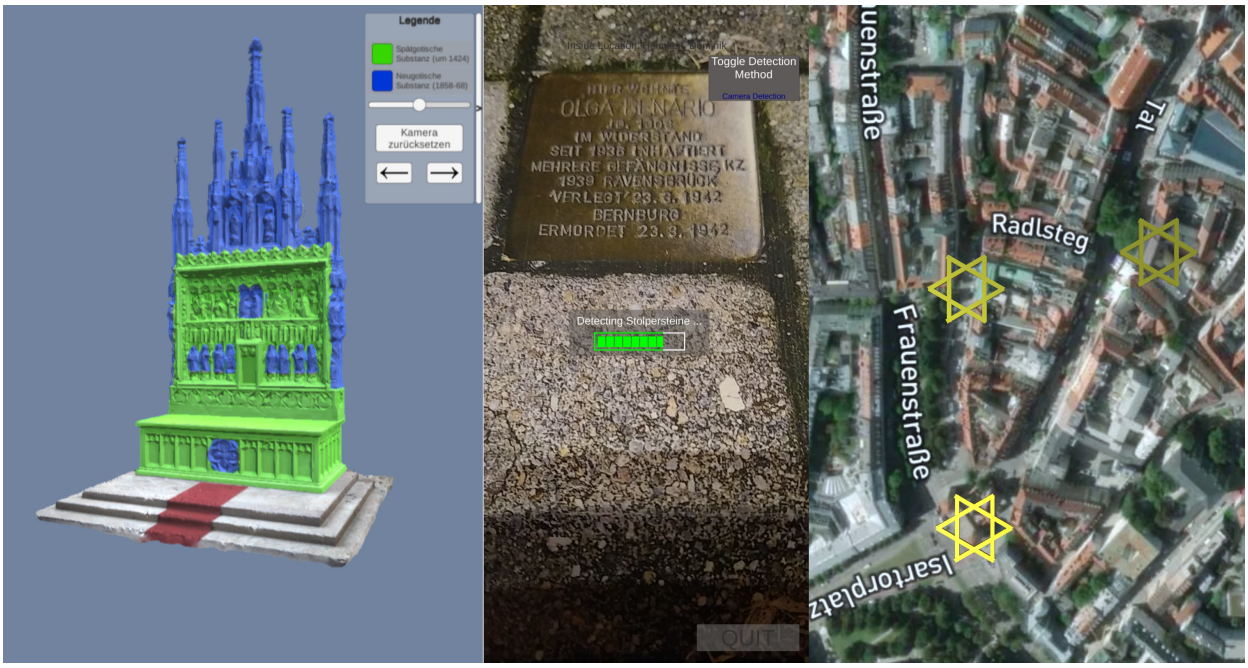

Fig. 1. Communicating cultural heritage in three different ways (from left to right): The altar of the St. Martin church in Landshut, the Stolperstein Memorial AR application, and the interactive map - Jewish Munich

Abstract—This lab-course explored the practical development and deployment of interactive applications, serious games, and augmented reality (AR) with the goal to raise awareness and teach about historical events, locations and buildings. This involved multiple projects, some of which were already in an advanced stage of development at the beginning of this lab-course. An example is the *StolpersteineAR* application, telling about the lives of holocaust victims, which has been developed in past research projects and was further improved by us based on additional user feedback. Expanding on the history of the Jewish people in Germany, we also developed an interactive map which allows historians to add important locations and information, and connect them based on historical data. This map along with its historical references can then be viewed by interested people. Apart from this, we worked on an interactive AR application, augmenting different components of a historical altar and making additional information, for example about inscriptions, available. Both an on-site and off-site (non-AR) version of this application is available.

Index Terms—Serious gaming, augmented reality, historical monuments, cultural heritage

1 Introduction

In historical research, apart from gathering and conserving information about historical events, sites, figures, and monuments, a key aspect is about how this acquired knowledge is provided and presented to the general public. This includes the teaching of history in schools and large, dedicated memorial sites or museums, but also the process of raising awareness about small pieces of history hidden throughout our day-to-day lives. Even though these areas pose similar challenges and can certainly have shared solution approaches, this research focuses on the last one: Improving the visibility, informativeness, and appearance of various historical sites and objects found outside of museums, for example on a public square.

We are convinced that several concepts borrowed from the field of research on video games can help with that. Serious games, for example, have been deployed in various other fields [15], but also teaching about cultural heritage specifically [11]. Moreover, already the technical possibilities alone can provide an enhanced experience, as for example old and decaying objects can be digitized and therefore made accessible for future generations. This can include for example 3D-scans [4, 12], digitization of historical documents [6], and recordings of eyewitness reports [10]. For example, of the Cathedral Church of Saint Peter in Cologne (ger. *Kölner Dom*) a 3D scan was performed and is now publically available for a virtual walkthrough[1].

With the 3D scanning market constantly growing [7, 8], the raw data required for digitization is easy to acquire for most artifacts and monuments. However, creating a meaningful, engaging, and informative interaction with those digital copies requires a considerable amount of work and resources as well. By implementing such applications and comparing them between each other, we were able to estimate problems and opportunities for future projects, which might scale up the amount of data presented and processed by such programs.

[1] https://koelnerdomlive.wdr.de/

Fig. 2. Snippet of the *Stolperstein*-table on *Wikipedia* regarding region Munich. On the *Wikipedia*-web page, the user can retrieve where the stones are placed, what they look like, the inscriptions of the stones and additional information about the lives of the persons to whom the stones are dedicated. (Source: [13])

Fig. 3. Screenshot of the AR application augmenting the *Siegestor* in Munich with the state during World War II in about 1944. (Source: [16])

2 Related Work

In addition to the classic methods of providing and presenting the acquired knowledge discussed in the introduction, there are also possibilities offered by new technologies. These can be, for instance, simple websites that conveniently make knowledge available to the general public over the internet. For example, *Wikipedia* provides a list of *Stolpersteine* for almost every region where *Stolpersteine* were placed.[2] [3]. The *Stolpersteine* are part of a big decentralized Jewish memorial to commemorate Holocaust victims. On the *Wikipedia*-web page, the user can retrieve which stones exist in the region, where they are placed, what they look like, the inscriptions of the stones, and additional information about the lives of the persons to whom the stones are dedicated (see figure 2).

However, the new technologies from computer science allow us to provide cultural heritage with even more advanced approaches than a static website. As part of a master's thesis[4], a framework was developed for presenting location dependent AR content in outdoor environments [16]. Thereby, the goal was to create a generic AR application that is independent of environmental factors such as different light or weather conditions, as well as not specifically tailored to certain building structures or properties [16]. The approach was tested on the *Siegestor*[5] in Munich where different states of the monument were augmented onto the actual structure (see figure 3). In our first project of this practical course, we take a similar approach, only less generic. In this project, the goal is to augment the altar of the St. Martin's Church in Landshut to make inscriptions readable and to visualize different building materials.

3 The Altar of the St. Martin's Church in Landshut

3.1 Historical Context

The St. Martin's Church (ger. *Martinskirche*) in Landshut, Germany (see figure 4), was built from around 1392 (earliest documentation of the construction) until 1500 [1]. In 1424, the first part of today's main altar (see figure 5) was constructed from sandstone [3]. Additional elements were added during the baroque period around 1664, yet those were removed later-on and are not part of the current altar construction anymore [2].

Still intact, however, are the neo-Gothic elements, which were added in the 19th century [3]. Therefore, the altar currently consists of parts which can be assigned either to the original altar from 1424 or the additions or restorations made during the 19th century (see figure 6). Another notable element of the altar are the many inscriptions which are placed on various locations on the altar. Those inscriptions contain bible citations and other texts, which are, however, hard to read for people visiting the church (see figure 7).

Fig. 4. The St. Martin's Church in Landshut, Germany, was constructed over roughly 120 years and eventually finished around 1500 [1]. (Source: [2])

3.2 Implementation

Considering the historical background of the altar and the many details incorporated in it, we focused on the two aspects described in section 3.1: The different parts corresponding to the year 1424 and the 19th century, respectively, and the inscriptions. The goal was to display those details in a clear and comprehensible way and give room to explore and analyze the altar's composition. Furthermore, we wanted to allow the user to use the concept of our application both at home as well as in an engaging way on-site, i.e., while standing in front of the altar. We therefore decided to develop a web-based program which works with an interactive 3D-representation of the altar[6] and, additionally, a mobile AR application, which attempts to track and augment the real altar. Both applications were implemented with the game engine *Unity*[7].

3.2.1 Data preparation

Since both of these approaches require a very accurate 3D model of the altar, we were provided with a mesh which was obtained through a 3D scan of the altar. We used this 3D model as a template to generate

[2] https://de.wikipedia.org/wiki/Liste_der_Stolpersteine_in_M%C3%BCnchen

[3] https://de.wikipedia.org/wiki/Liste_der_Stolpersteine_im_Landkreis_F%C3%BCrstenfeldbruck

[4] https://wiki.tum.de/display/infar/MA%3A+Siegestor+AR

[5] https://en.wikipedia.org/wiki/Siegestor

[6] The resulting application can be opened via this link: https://levermington.itch.io/landshut-altar

[7] https://unity.com/

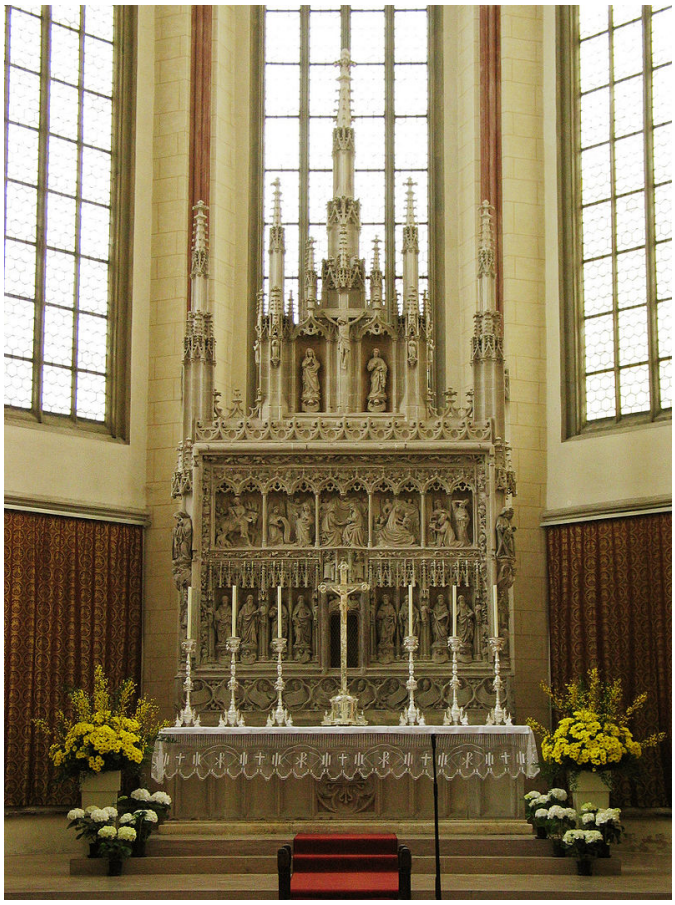

Fig. 5. The main altar of the St. Martin's Church in Landshut, Germany. The basis was built during the construction of the church itself. Multiple times in history, the altar's appearance was changed by adding or replacing parts. Especially, the changes during the neo-Gothic period are still reflected in the altar's present form. (Source: [2])

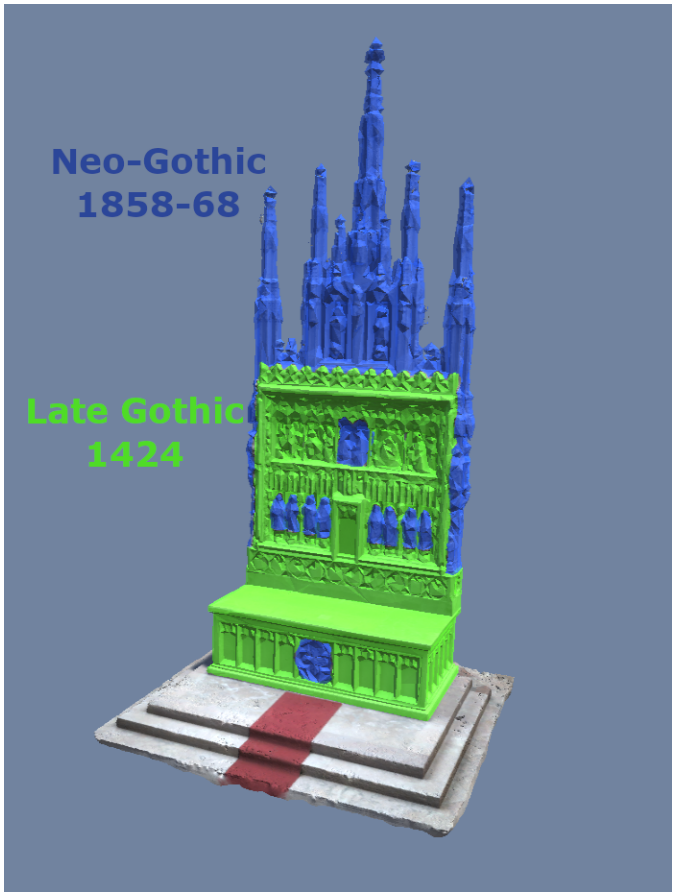

Fig. 6. We created a custom, semi-transparent 3D mesh (here displayed with an alpha value close to the maximum) which acts as a colored overlay for the 3D model of the altar. With its two colors, it highlights which parts of the altar originate from the original construction from 1424 (green) and which parts were added in the neo-Gothic period (blue).

Fig. 7. The altar's ornaments are embellished with a variety of chiseled scrolls, most of which have texts engraved into them. Nowadays, these texts are difficult for visitors to read.

Fig. 8. Each inscription on the altar received a custom 3D mesh. Those were then used to detect the user's interactions and highlight the inscription on the corresponding location.

and manually adapt a new mesh, which enveloped the original mesh very tightly. We then separated this mesh into the subcomponents of the altar, which correspond to the original construction and the neo-Gothic additions, respectively (see figure 6). By coloring the submeshes differently, this allowed us to visually represent the different subcomponents as an overlay with a user-defined alpha value, so that the original altar texture can still be visible.

Furthermore, we wanted the user to be able to discover and interact with the inscriptions incorporated into the altar. For this purpose, we manually created a 3D mesh for each inscription, which mimics the 2D shape of each inscription on the altar respectively and extrudes it along the approximate surface normal into a 3D volume (see figure 8). Those volumes are then later-on used to mark and highlight the corresponding inscription on the 3D model or the augmentation.

3.2.2 Web-based program

For the web-based approach, we directly used the 3D model from the 3D scan as basis. This model can be rotated and zoomed into, which allows the user to see it from all possible angles. However, when testing this application the feedback indicated, that users, especially with little prior knowledge of 3D programs, might get confused by the rotation mechanic and be unable to find their way back from a more complicated rotation. We therefore added another and more simplistic rotation mechanic via GUI buttons. It also included the option of resetting the orientation to reach new angles easier.

For displaying the subcomponents of the altar, we added two GUI labels, which correspond to the two colors, green and blue, and displayed the historical data about them, i.e., the late Gothic substance from 1424 and the neo-Gothic substance from 1858-68, respectively (see figure 10). By clicking on one of them, the corresponding overlay is turned on and off. An additional slider controls the transparency of both overlays, as long as they are turned on.

In order to allow interaction with the inscriptions, we added a simple collider to each inscription. This was made easier by the fact, that each inscription was represented with an individual mesh object, as this allowed the game engine to adjust the corresponding collider, in our case a sphere collider, automatically to the extensions of the inscriptions. What, however, a disadvantage of this approach is and what alternative we had, will be discussed in section 3.2.4.

The colliders then allowed us to test the mouse position against the projected inscriptions, which we used in order to highlight inscriptions the user's mouse hovered over and to make them clickable. When clicking on an inscription, its text will be displayed as a GUI text field (see figure 9).

3.2.3 Mobile AR application

The mobile AR application utilizes the altar model from the 3D scan in order to create a tracking target with the AR platform *Vuforia*[8] in combination with Unity. Furthermore, we used the model to create a

[8] https://www.ptc.com/en/products/vuforia

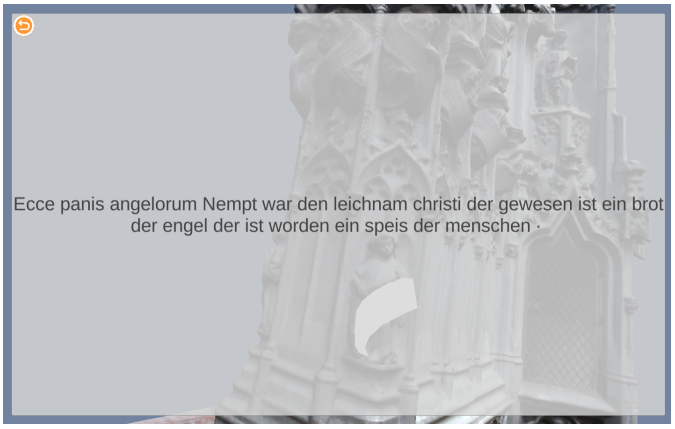

Fig. 9. Any inscription is highlighted when hovering over it with the cursor, which can be slightly seen in the background. It can then be clicked, which displays its context in a text field.

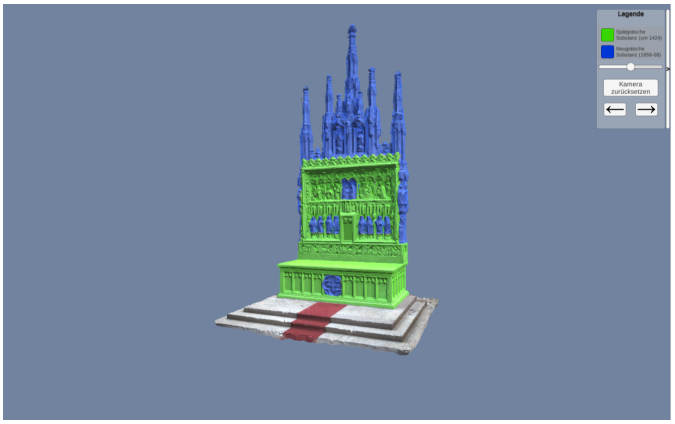

Fig. 10. Our web application runs via *WebGL* and can be embedded in websites. It is meant to provide easy access and usage from the user's home PC.

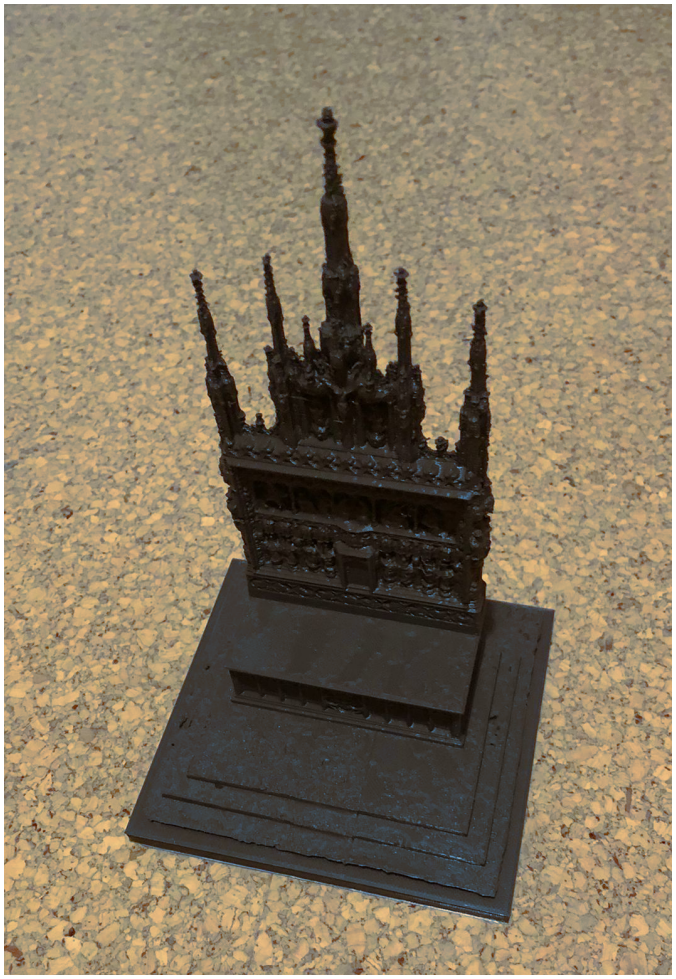

Fig. 11. In order to test the AR tracking of our application during development, we 3D-printed a miniature model of the altar. This 3D-print has an approximate size ratio of 1:35 compared to the original altar.

3D-printed of the altar (see figure 11), with which we tested our AR application (see figure 12).

Compared to the overlay in section 3.2.2, the AR application provides a very similar interface to the user with almost identical functionality and interactions. The inscriptions, however, are being read differently: Because of the inherent instability of a hand-held mobile device, which we have to assume for our use-case, it is very unlikely that a tap on the display allow a precise selection of the correct inscription. Therefore, we provide the user with two additional GUI buttons, which go back and forth through a list of all inscriptions. The currently selected inscription is then highlighted in the augmentation (see figure 13).

3.2.4 Reusability

What became apparent during our project, was that much of our effort was spent on features and components, which were tailored to the specific instance of cultural heritage and would therefore have to be repeated for, for example, every altar, for which such an application should be developed. An example for this are the inscriptions, of which every single one corresponds to a custom 3D model. To go through the same effort of modeling 3D models for every significant element of multiple of such digital models, would just not be feasible. We therefore tried to think of alternative approaches, which might allow the historian familiar with the historical background themselves to add the necessary information easily. Ideally, a pipeline would exist which simplifies and streamlines the process to, for example, separate the model into different components which are then assigned to the corresponding

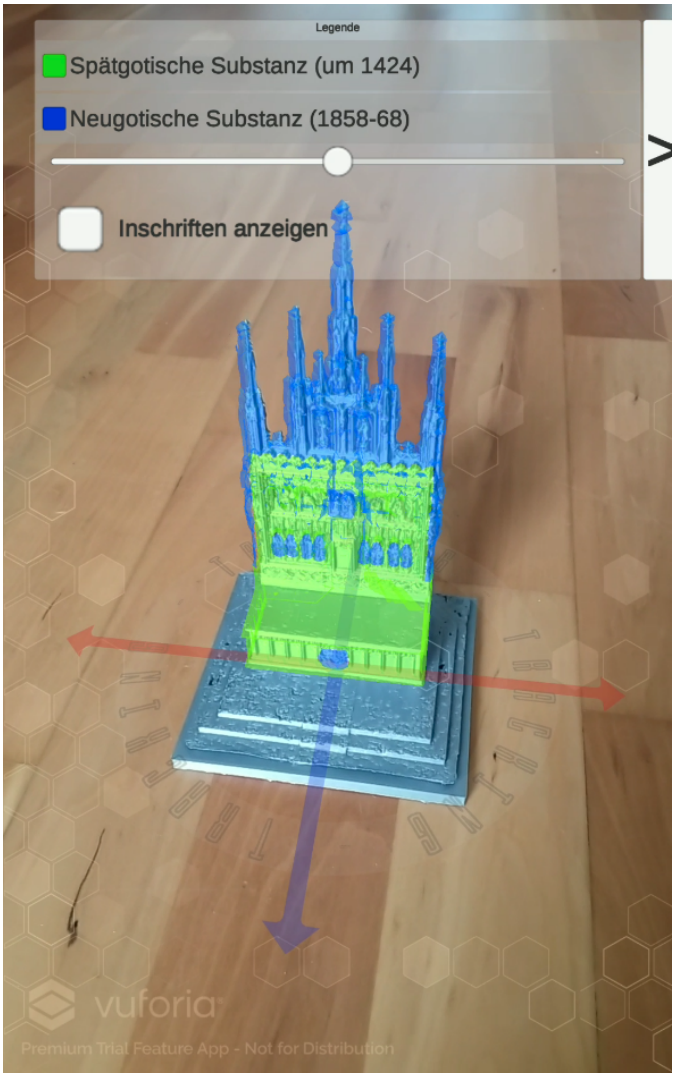

Fig. 12. Our application can successfully detect, track, and augment the miniature 3D-print of the altar. In this picture, we used a print made from a lighter material in order to improve the tracking performance.

Fig. 13. By providing the user with two buttons to go through all inscriptions, we avoid inaccurate screen interaction in order to select the correct inscription. Of course, at the same time, the user might have to skip through many unwanted inscriptions before arriving at the desired one. It should be noted, however, that many inscriptions are related to the preceding one, and we tried to maintain that order in our application.

Fig. 14. *Stolpersteine* of the *Stolperstein-Memorial* (Source: [14])

historical eras. In our case, this could have been theoretically achieved by providing a painting tool via the backend to the historian, who would then proceed to color the respective surface areas and assign each color to the corresponding time. However, since the process of developing the drawing tool itself was not within the scope of this research, we attempted the manual, 3D-based realization. For this single altar, this was the easiest and probably the fastest approach for developers familiar with 3D software.

4 Stolperstein Memorial AR-App

4.1 Concept

In the second project, augmentations were also used - similar to the altar-project - to present additional, historical information at specific locations, more precisely on *Stolpersteine*. As part of the *Stolperstein-Memorial*[9], *Stolpersteine* are metal paving stones on which short information about national socialism victims are engraved (see figure 14) [5]. Such a stone is placed at the location where the last self-chosen home of the victim was [5]. However, only short information about the holocaust victims can be depicted. To address this shortcoming, two bachelor thesis[10] [11] have implemented an AR-application, that augments the *Stolpersteine* with much more detailed data than what can be read on the stone. Additionally to the name, date of birth and death, and where the person died, pictures, sound recordings, relationships, life milestones, and much more additional information can now be accessed with this mobile-app (see figure 15).

4.2 Increase the Usability

While we were testing the app from the previous thesis, we noticed that the recognition of the stones takes quite a long time. In a next step, we wrote a code that told us how long it exactly takes to detect the stone. Surprisingly, we realized that it only needs a few seconds. We came to the point that it only feels subjectively long, as the user gets no feedback that the app is still processing. So, we decided to add a loading bar, what can be seen in figure 16. In general, we were able to determine in this project through discussions with users that the usability still needs to be increased to create a beneficial and user-friendly application, especially for people with little affinity for IT. It is usually the case that computer scientists write programs used by groups that have other priorities than pure functionality, what in turn is often the main requirement for computer scientists. Due to these shifted interests, different priorities, and different understandings of technology, a constant exchange between developers and later users is mandatory during the development process.

[9] https://www.stolpersteine.eu/

[10] https://wiki.tum.de/display/infar/%5B21SS+-+BA%5D+Development+of+an+Augmented+Reality+App+for+Stolperstein+Memorials

[11] https://wiki.tum.de/display/infar/%5B22WS+-+BA%5D+Development+of+a+User+Friendly+Content+Creation+System+for+the+Stolperstein+Memorial+Augmented+Reality+Application

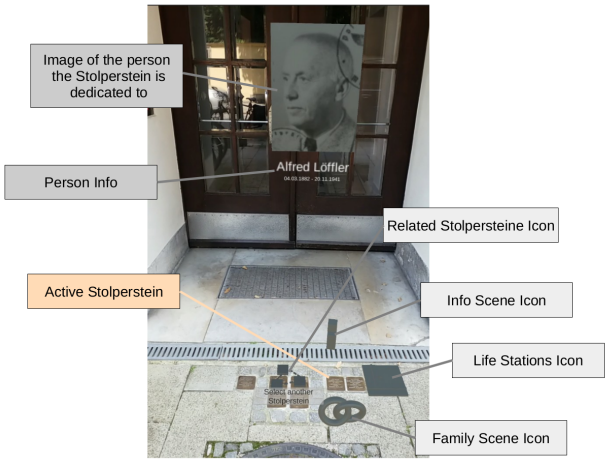

Fig. 15. Explaining screenshot of the Stolperstein AR-application (Source: [9])

Fig. 16. Screenshot of the Stolperstein AR-application while detecting a *Stolperstein*

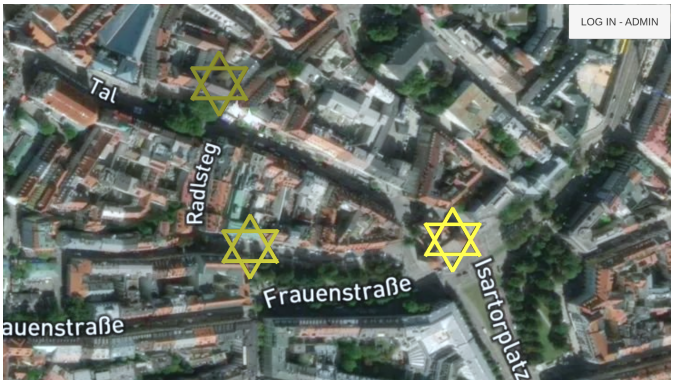

Fig. 17. Screenshot of the interactive map: *Stars of David* are placed to mark locations with Jewish history

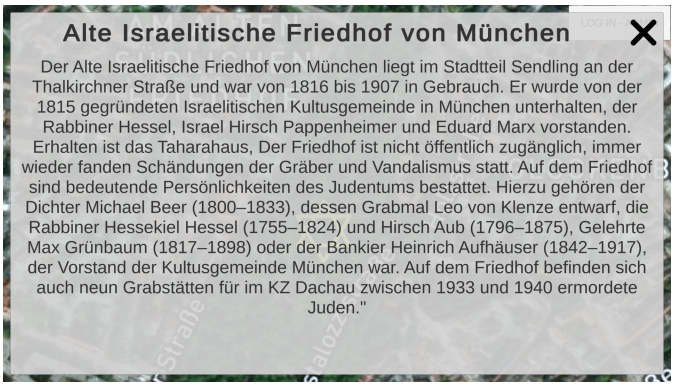

Fig. 18. Screenshot of the interactive map: After clicking on a *Star of David*, information about the respective place is displayed

5 Interactive Map - Jewish Munich

5.1 Concept

In a collaboration with the history faculty of the *Ludwig-Maximilians-Universität München*[12], we were looking in our third project for a way to present information about local places in Munich with a Jewish background. Since the content is about local places that you might pass in your daily life, the main objective is to convey expert knowledge to the general public. Therefore, the challenge was to choose a format that would transform the dry, factual, and sometimes unprocessed materials of the history faculty into something that would be accessible and appealing to the general public and not just to experts in the field. At this point, games engineering comes into the picture. We decided to implement a *Google maps*-like interactive map running on a website[13] because this format is widely used and understood by the general public. To realize it, we use the *mapbox Maps SDK for Unity*[14]. On top of the map, *Stars of David*[15] are placed as markers of specific locations with Jewish history (see figure 17). By clicking on them, information about the Jewish history of the respective place is displayed, what can be seen in figure 18.

5.2 Authentication of Admin Users

In a next step, we had to provide a way for historians to dynamically incorporate new information in the map. However, the backend must also be kept intuitive, as it will also be used by people unfamiliar

[12] https://www.geschichts-und-kunstwissenschaften.uni-muenchen.de/index.html

[13] Interactive Map - Jewish Munich: https://huberdo1909.itch.io/interaktive-karte-jdisches-mnchen

[14] https://docs.mapbox.com/unity/maps/guides/

[15] https://en.wikipedia.org/wiki/Star_of_David

with computer science. Moreover, as Jewish history is a sensitive topic, it is important that only authorized persons are allowed to change data. Therefore, we decided to have an admin area that can only be accessed by authorized persons. Here, you can create new admin-user accounts, add and delete new markers and update their information. For the authorization process, we use *Firebase*[16], which provides an database managing the users with their hashed password, email and idtoken. With the help of a post-request to the address *https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword* respectively *https://identitytoolkit.googleapis.com/v1/accounts:signUp* of the *RestClient*-library[17] [18], we were able to sign in users and also create new ones. To ensure security, the authentication key of the *Firebase*-project is required for the post-request.

The code-snippet below shows the most important parts of code for signing in a user. The variable *user* is an object of the *User*-class, managing the email, password, and if an idtoken and an update token should be returned or not. *AuthKey* is the authentication key of the *Firebase*-project and *SignResponse* is an object of the *SignResponse*-class, which holds the returned idtoken of the request.

```
User user = new User(email, password);

RequestHelper currentRequest = new RequestHelper
{
    Uri = "https://identitytoolkit.googleapis.com/v1/
    accounts:signInWithPassword",
    Params = new Dictionary<string, string> {
            { "key", AuthKey}
        },
    Body = user,
};

RestClient.Post<SignResponse>(currentRequest).
Then(...).Catch(...);
```

5.3 Admin Area to Manage the Dynamic Data

As mentioned above, new markers, new clickable *Stars of David*, can be created. To do so, admin-users enter the title, the content to be displayed, and the location, as shown in figure 19. To simplify finding the latitude and longitude of the location, a link to a web-tool[19] has been integrated, with which the respective latitude and longitude can be output both by clicking on a map and by entering an address. Furthermore, already existing markers can be modified and also deleted, what can be seen in figure 20. In principle, the user has similar input methods as when creating markers, except that he or she now additionally selects in the drop-down menu which marker he or she wants to delete or update.

5.4 Technical Aspects of the Admin Area - Server Communication

In the background we are again helped by *Firebase*[20] and *RestClient*, which also support a realtime database and a server-client structure. The database is structured as a key-value pair with the key *MarkerData* and an array of other key-value pairs as value. The elements of the array consist of an id as key and a c#-object as value. The structure can be seen in figure 21. When retrieving or sending data to the database, we use the *JSON* file format[21]. To update existing data or create a new marker, we send via put-request the new or updated *MarkerData*-object, containing the id, title, content, latitude and longitude of the

[16] https://firebase.google.com/docs/reference/rest/auth?authuser=0

[17] https://assetstore.unity.com/packages/tools/network/rest-client-for-unity-102501

[18] https://github.com/proyecto26/RestClient

[19] https://www.revilodesign.de/tools/google-maps-latitude-longitude-finder/

[20] https://firebase.google.com/docs/reference/rest/database?authuser=0

[21] https://en.wikipedia.org/wiki/JSON

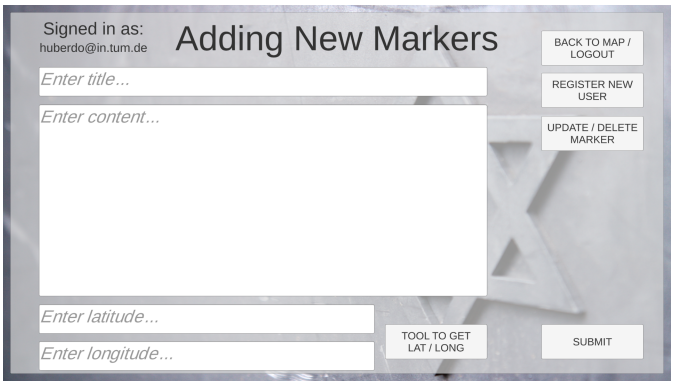

Fig. 19. Screenshot of the interactive map: Admin area, where new markers can be added

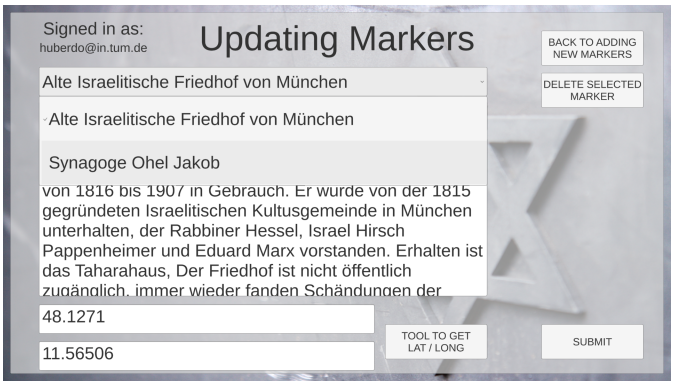

Fig. 20. Screenshot of the interactive map: Admin area, where existing markers, selected in the drop-down menu, can be updated and deleted

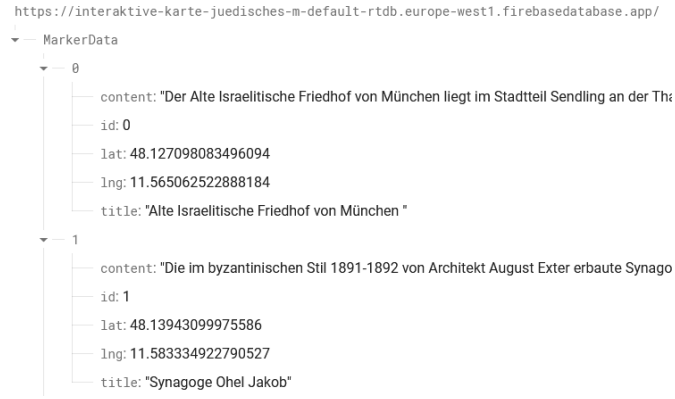

Fig. 21. Screenshot of the interactive map database structure

marker, to the url-address *<url of the project >/MarkerData/<id of the marker >.json?auth=<idToken of the signed-in user >*. This request will change the values of the key-value pair, so change the *MarkerData* c#-object, in the array at the respective key-index or will create at this spot a new key-value pair if none exists yet. Below, the code of the put-call is shown:

```
RestClient.Put(url + "MarkerData/" + id +
".json?auth=" + idToken, new MarkerData(id, title,
content, lat, lng)).Then(...).Catch(...);
```

When deleting a marker, we just delete the key-value pair from the array in the database. This can simply be done with

```
RestClient.Delete(url + "MarkerData/" + id +
".json?auth=" + idToken).Then(...).Catch(...);
```

where *url* is the address of the project, *id* is the id of the marker that should be deleted and *idToken* is the idtoken of the currently signed-in user.

All dynamic data of the interactive map is only loaded once in the beginning, when the map is started. After that, only a reload must take place after the data has been updated. To retrieve data from the database, a *GetArray-RestClient* request is executed. This request is to the main part a simple get-http request but with the particularity that an array is returned. This is very beneficial for our array-based database structure, since it allows us to get the whole content as an array of *MarkerData*-objects with just a single get-request. However, since we simply delete entries of the array in the database when deleting a marker and then get the whole array using a GetArray-request, the received array has empty cells, which is impractical for later operations. Therefore, it is important to compress the returned array after the GetArray-request, i.e. to put the data in a smaller array, with no empty cells. This and the general GetArray-call can be seen in the code snippet below.

```
RestClient.GetArray<MarkerData>(url + "MarkerData" +
".json").Then(response =>
        {
            // filter response for all empty spaces
            //in the array
            int nullCounter = 0;
            for(int i = 0; i < response.Length; i++)
            {
                if(response[i].title == null)
                {
                    nullCounter++;
                }
            }
            if (nullCounter == 0)
            {
                (...);
            }
            else
            {
                MarkerData[] smallerArray = new
                MarkerData[(response.Length -
                nullCounter)];

                for(int i=0, j = 0; i <
                smallerArray.Length; i++, j++)
                {
                    while (response[j].title == null)
                    {
                        j++;
                    }
                    smallerArray[i] = response[j];

                }
                (...)
            }
                (...)
        }).Catch(...);
```

In order to ensure that the additional effort caused by moving the data arises rarely, i.e. to have empty array-cells as seldom as possible, we try to close the holes in the array when inserting new data into the database. Therefore, we are deliberately inserting the new data into the empty cells of the *MarkerData*-array, i.e. assigning the marker-id accordingly before calling the put-request. Hence, we always look for the first unallocated id. The code-snippet below shows how exactly the id is set. First, we get a current version of the database content. If this content is null, the database-array is empty, and we can create the *MarkerData*-object with id 0, so on index 0 in the array. If not, we set the id to the length of the received array, so we put the new *MarkerData* at the end of the array. In the next step, we proceed through the obtained array and check if an id of a *MarkerData*-object is not equal to the array index, which would mean that there is a hole in the array at this particular index and thus no *MarkerData*-object with this index value as its id exists. So we choose this index as the id for the new *MarkerData*-oject, which therefore fills the hole in the array.

```
// Get current database to find the id where to put
// the new marker data
StartCoroutine(DatabaseManager.RetrieveFromDatabase
(markerdataArray =>
{
    if(markerdataArray == null)
    {
        // the database is emtpy:
        id = 0;
    }
    else
    {
        // set id to last id+1 --> id gets overwritten,
        // if an id is missing
        id = markerdataArray.Length;

        for (int i = 0; i < markerdataArray.Length; i++)
        {
            if(markerdataArray[i].id != i)
            {
                id = i;
                break;
            }
     1   }

        StartCoroutine(DatabaseManager.PutToDatabase(
        id, title.text, content.text, latFloat,
        lngFloat,(...)));
    }
}));
```

6 Conclusion

Whether applications such as ours are really applicable as a more engaging and informative way of conveying information about our cultural heritage, though part of the underlying motivation, must remain the focus of other research. Rather than that, our research investigated possible chances and challenges for developing serious games in the context of cultural heritage. To that end, our findings show that practices borrowed from the game industry can be applied to cultural heritage in various ways. Especially, the field of AR can open new ways to combine the vividness of real objects with the advantages of adding and displaying information digitally.

However, our projects also highlighted a major drawback for the combination of serious games and cultural heritage. The lack of a unified, streamlined pipeline for preparing the data, even if available, makes the process of developing cultural heritage applications very cumbersome and on a large scale not feasible. We therefore conclude, that it would benefit both the programmer and the historian in their cooperation, if

the programmer concentrates on providing an easy-to-understand and easy-to-user back for a larger set of historical artifacts, which is then populated with data by the corresponding historian. Hence, for every altar, for example, a historian would just need the 3D object, upload it, and have some sort of tool to mark and annotated the 3D structure. In theory, the rest of such a backend could then be automatized.

7 Outlook

Thinking about potential next steps for our projects, we strongly recommend conducting evaluations and studies. This would be feasible through user surveys and expert reviews, since our applications are both intended for interested, common people and also historians, which provide the information. Based on the user feedback, we should further improve the usability and fill it with more data, especially the interactive map. This project could also be adapted to be used for other cities, or the altar project could be generalized to support other objects, like another altar from another church. Furthermore, builds for mobile platforms could also be a next step for the altar-application and the interactive map, which currently are only *WebGL*-builds on *itch.io*.

In the big picture, as mentioned in section 3.2.4 and 6, we should come to the point to have more generalized software to provide cultural heritage. Easy backend interfaces, with which historians can provide cultural heritage information, as well as generalized pipelines could help to minimize the needed time to implement applications that depict this information and to achieve more generic software that can be used for multiple fields of use in communicating cultural heritage and more general history.

Acknowledgments

The authors wish to thank the historians of the *Ludwig-Maximilians-Universität München*, especially Prof. Dr. Bernd Päffgen. We would also thank all people who maintain and attend to the St. Martin's Church in Landshut, as well as all other instances of cultural heritage.

References

[1] Geschichte - pfarrei st. martin landshut. Available at https://martin-landshut.de/st-martin-besuchen/geschichte/, 28 Sep 2022.

[2] Martinskirche (landshut) - wikipedia. Available at https://de.wikipedia.org/wiki/Martinskirche_(Landshut), 28 Sep 2022.

[3] Sehenswürdigkeiten - pfarrei st. martin landshut. Available at https://martin-landshut.de/st-martin-besuchen/sehenswuerdigkeiten/, 28 Sep 2022.

[4] G. Bogdanova, T. Todorov, and N. Noev. Digitization and 3d scanning of historical artifacts. *Digital Presentation and Preservation of Cultural and Scientific Heritage*, 3:133–138, 2013.

[5] Demnig, Gunter and Heuper, Jérôme. Stolpersteine. Available at https://www.stolpersteine.eu/, 29 Sep 2022.

[6] B. Endres. Digitizing medieval manuscripts. In *Digitizing Medieval Manuscripts*. ARC, Amsterdam University Press, 2019.

[7] Grand View Research Inc. 3d scanning market size, share trends analysis report by product (optical scanner, laser scanner), by range, by end-user, by component, by type, by application, by region, and segment forecasts, 2019 - 2025. Available at https://www.grandviewresearch.com/industry-analysis/3d-scanning-industry, 20 Sep 2022.

[8] H. Jangra, T. Upadhaya, and V. Kumar. 3d scanning market by type (optical scanner, laser scanner, and structured light scanner), services (reverse engineering, quality inspection, rapid prototyping, and face body scanning), range (short range scanner, medium range scanner, and long range scanner), and application (entertainment media, aerospace defense, healthcare, civil architecture, industrial manufacturing, and others): Global opportunity analysis and industry forecast, 2021-2030. Available at https://www.alliedmarketresearch.com/3D-scanning-market, 20 Sep 2022.

[9] Lenßen, Nicolas. Development of an augmented reality app for stolperstein memorials. Available at https://wiki.tum.de/display/infar/%5B21SS+-+BA%5D+Development+of+an+Augmented+Reality+App+for+Stolperstein+Memorials, 26 Sep 2022.

[10] M. Ma, S. Coward, and C. Walker. Question-answering virtual humans based on pre-recorded testimonies for holocaust education. In *Serious games and edutainment applications*, pp. 391–409. Springer, 2017.

[11] M. Mortara, C. E. Catalano, F. Bellotti, G. Fiucci, M. Houry-Panchetti, and P. Petridis. Learning cultural heritage by serious games. *Journal of Cultural Heritage*, 15(3):318–325, 2014.

[12] M. Pieraccini, G. Guidi, and C. Atzeni. 3d digitizing of cultural heritage. *Journal of Cultural Heritage*, 2(1):63–70, 2001.

[13] Schikora, Andreas et al. Liste der stolpersteine in münchen. Available at https://de.wikipedia.org/wiki/Liste_der_Stolpersteine_in_M%C3%BCnchen, 29 Sep 2022.

[14] Stadtarchiv Münster. Erinnern nach 2000 - projekt "stolpersteine" des künstlers gunter demnig sowie verein "spuren finden e.v.". Available at https://www.stadt-muenster.de/kriegerdenkmale/erinnern-nach-2000/projekt-stolpersteine, 26 Sep 2022.

[15] T. Susi, M. Johannesson, and P. Backlund. Serious games: An overview. 2007.

[16] Tolstoi, Paul. Siegestor ar. Available at https://wiki.tum.de/display/infar/MA%3A+Siegestor+AR, 29 Sep 2022.