Augmented and Virtual Reality Object Repository for Rapid Prototyping

Ivan Jovanovikj, Enes Yigitbas, Stefan Sauer, Gregor Engels

Augmented and Virtual Reality Object Repository for Rapid Prototyping

Ivan Jovanovikj[0000-0002-1838-794X], Enes Yigitbas[0000-0002-5967-833X], Stefan Sauer[0000-0003-3084-0409], and Gregor Engels[0000-0001-5397-9548]

Department of Computer Science, Paderborn University, Germany
{firstname.lastname}@upb.de

Abstract Augmented Reality (AR) and Virtual Reality (VR) are technologies on the rise and as their market grows, speeding up the process of developing new ideas becomes even more important. In achieving rapid development, rapid prototyping plays a very important role. To support rapid prototyping, this demo paper presents an approach that relies on an object repository. It enables scanning, editing, storing and publishing of virtual objects that can be reused in different augmented and virtual reality applications. To show the benefits of the approach, an exemplary scenario is illustrated by prototyping an interior design application.

Keywords: augmented reality, virtual reality, object repository, rapid prototyping

1 Introduction

Augmented Reality (AR) and Virtual Reality (VR) have received grown interest by customers and media. The worldwide spending is predicted to over 20 billion USD in 2020[1] and up to 192 billion USD in the following three years[2]. Hence, the rapid development of new applications becomes even more important. Therefore, software designers and engineers should be able to quickly build AR/VR prototypes. However, developing such prototypes is not an easy task as the existing tools and SDKs are often highly technical and as such require much up-front learning effort [2,5]. Additionally, they are platform-dependent, making platform evaluations more complex. To support the designers in reusing existing objects in multiple prototypes and to gain a quick look at how the application might look and work, a large collection of different kinds of objects is needed. A fast way of building such a collection is translating physical objects into virtual ones. To speed up the prototyping of AR and VR applications, we combine an object repository with prototyping mechanisms. Therefore, in this demo paper, we present *AVROsitory* which is an **A**ugmented and **V**irtual **R**eality **O**bject Repo**sitory**. Our approach utilizes the strengths of both object repositories and prototyping tools and comprises three main components: server (with an object repository), mobile client, and a web client component, as shown in Fig. 1.

[1] idc.com `https://bit.ly/3cI02HT` [2] statista.com `https://bit.ly/3jgLX6x`

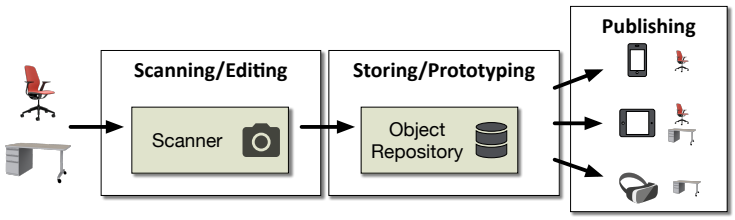


Fig. 1: Overview of the solution idea

The *Scanning/Editing* component is a mobile client which enables the designers to add new objects or edit existing ones. The real-world physical objects can be scanned and transferred into digital objects via Photogrammetry [1]. It is also possible to scan physical objects with a smartphone and add them to the repository as virtual objects. Photogrammetry offers a quick solution for creating mock-ups, that can be used for prototypes, but as final AR/VR objects too. The *Storing/Prototyping* component, i.e., server, is a an object repository which enables reusability by providing a tagging system to classify the objects, thus allowing a quick search on existing objects. Furthermore, objects can be also classified by the level of detail quality so that depending on the usage scenario an object with a suitable quality can be provided. While in some cases an object must be in great detail, the same might be used in the background somewhere else, requiring less detail and fewer resources. Business logic can be also added to objects with a specific task by using the logic templates. Finally, the *Publishing* component is a web client which provides a mechanism (Unity Scripted Importer) to Unity IDE. As a result, a reusable Unity object (prefab) is added to the project's assets in Unity and it can be used for developing AR/VR prototypes or AR/VR applications. Furthermore, it provides also an option for creating custom importers for other IDEs.

The rest of the paper is structured as follows: In Section 2, the architecture of the repository is presented. To show the feasibility of our approach, in Section 3, we present an application example where the developed repository was used in a prototyping scenario. In Section 4, we briefly discuss the related work and in the end, Section 5 concludes the work and gives an outlook on future work.

2 Solution Concept

In this section, we present the solution concept which builds upon an object repository for prototyping and development of AR/VR applications. As shown in Fig. 2, our solution consists of three main components. The central component is the server which contains the object repository.

Before explaining the object repository, we shortly discuss the different types of supported objects. An *object* is defined as a 2D or 3D graphical model with optional business logic. It can depict small single real-world items as well as entire

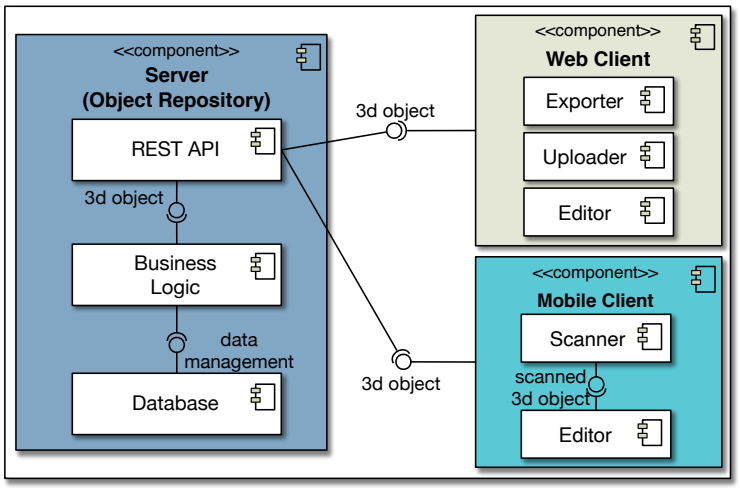


Fig. 2: Solution architecture: A central server application with a web interface and an Android client.

rooms. The business logic is rather basic as it does not contain any application-specific context. For example, an object representing a button would contain only a basic on-click listener without specific behavior. By evaluating existing AR applications publicly available, we classified the objects based on the user interaction into *non-interactive*, *click/touch-triggered*, and *range-triggered*. A *non-interactive* object entails no business logic, e.g., a decorative object. On the contrary, a *click/touch-triggered* entails a business logic, e.g., a UI button. Finally, a *range-triggered* object is triggered according to a user's distance, e.g, a descriptive text appears when a user is in a given range. The object repository includes business logic for these object classes in terms of logic templates only, that can be written by the repository users, extending the existing solutions to their needs. So, the object repository stores graphics, business logic, and metadata. The objects are accessible via a REST-API which means that data is stored as entities, which can be retrieved, updated, and deleted via HTTP requests. The entities consist of metadata and binaries. Whereas the metadata is stored in a database, the binary objects are stored in the server's file system. The binary objects include original graphics, as well as converted binaries in formats distributed via the REST-API. The second component of the object repository is a mobile application which is the main client for designers. A mobile application was chosen, because of the wide availability of smartphones today, which makes prototyping easier and quicker. As the object repository is based on a server that offers an API, other client applications that run on Microsoft's HoloLens[3], or photogrammetry equipment, e.g., tablets, can be added later on. The mobile client supports multiple tasks. As the client application synchronizes with the server, the users can display and browse the repository everywhere. This enables

[3] `www.microsoft.com/hololens`

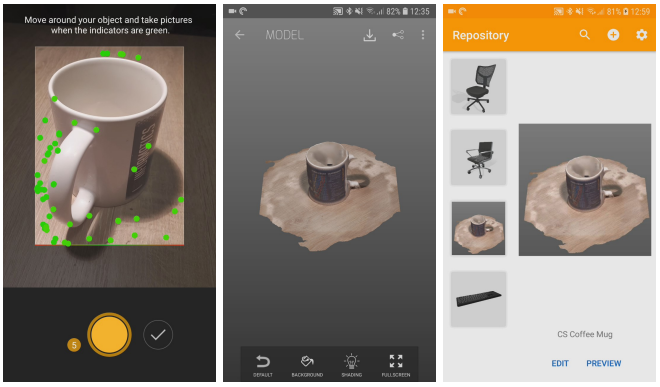


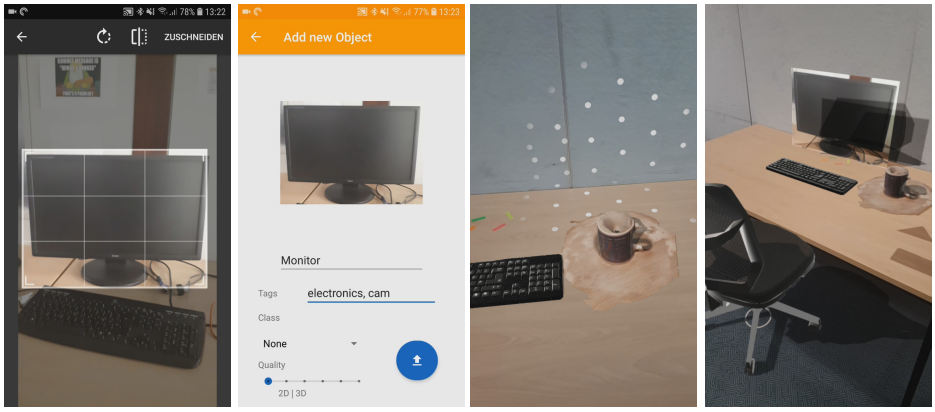
Fig. 3: Scanning of a coffee mug: Defining photogrammetry key points marked in green (left and middle); The imported object in the repository (right).

the users to preview their objects in their current environment. Furthermore, multiple objects can be combined into one composition thus allowing more complex mock-ups. The mobile client also enables users to create new objects and upload them to the server. The user can use photogrammetry and chroma-keying to create 3D and 2D graphics respectively. The objects can then be extended with metadata, like tags and quality levels. The last component is a web client that has similar functions to the mobile client, except that it does not allow any kind of scanning. However, it allows existing graphics to be uploaded. Its main purpose is to enable exporting objects to an IDE.

3 Application Example

This section presents an exemplary prototyping scenario of an interior design application. The application should enable users to decorate an empty office desk, to make design choices. Therefore, the application enables them to augment their desk with decorations and items from a catalog in the AR application. To achieve this, we firstly scan physical objects from an existing, real office desk and add them to the repository. These objects are then previewed, provided with business logic, if necessary, and in the further development process, replaced by high-quality 3D models. At the beginning, we have two 3D models of office chairs and one of a keyboard. Then, the three objects are added, all of them in good quality without any logic (classes). We begin by determining which catalog items we want to provide with our application. For simplicity, we support a basic set of coffee mugs, keyboard, monitor, and chairs. As our repository already contains chairs and a keyboard, we want to add models of coffee mugs and a monitor. The coffee mugs are scanned with photogrammetry, as shown in Fig. 3. As *AVROsitory* does not include an integrated photogrammetry solution, we

use Scann3D[4], a third-party tool. As we are in an early development phase, it is sufficient for our monitor model to be two-dimensional. Therefore, we are using the build-in object creation by taking a photo. We also provide metadata for the object, as shown in Fig. 4a, for further reuse.



(a) Cropping and importing of a photo in the repository

(b) Plane detection (left); Finished preview with simulated shadow (right)

Fig. 4: Cropping, importing, and previewing of repository objects

So far, we have all objects in our repository that we need. We can thus proceed to test how our application would look like, when the user previews the decoration. To do that, we create a composition of our repository objects (the screenshot on the right in Fig. 4b). To enable interaction with the objects, we firstly sketch a button prototype on a board and we add it to the repository by simply taking a picture, as shown in the screenshots in Fig. 5. After minor editing, the button can be used in the example application. To really test the added objects, the programmers create a prototypical application using the objects. Firstly, they import the objects into Unity using the *AVROsitory* web client and add some business logic to our buttons, as shown in Fig. 6.

As the development is going on, we design high-quality models for our furniture, and we add these models to our repository as high-quality versions of our prototyping objects. Also, more buttons are needed and for this purpose, we can reuse the existing button logic in the repository. As we now have higher quality objects, we can replace the mock-ups in the actual application, as shown in Fig. 7.

[4] Scann3D by SmartMobileVision. Paid application. Not related to this work.

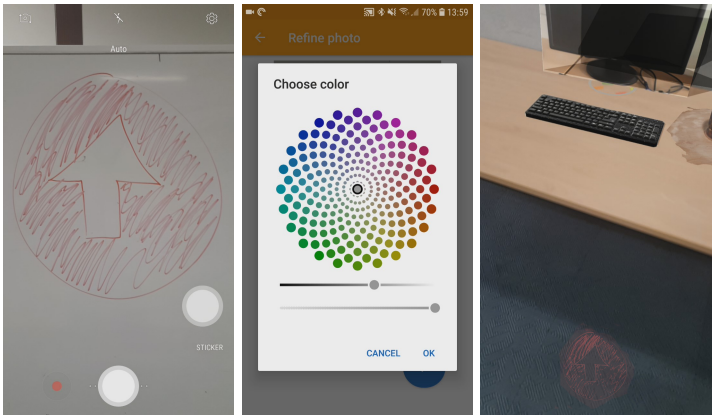


Fig. 5: Taking a picture in front of a monochromatic background (left) and using chroma keying to get a transparent background (middle and right).

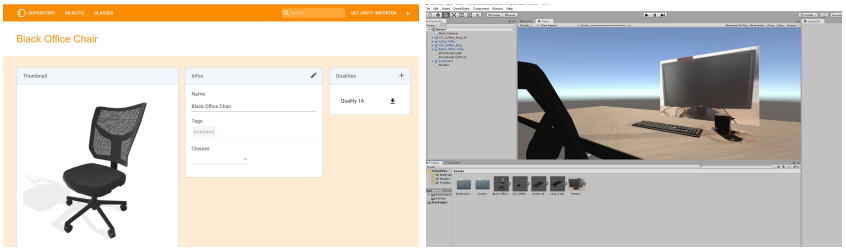


Fig. 6: Using the web client (left) to export our objects into Unity (right)

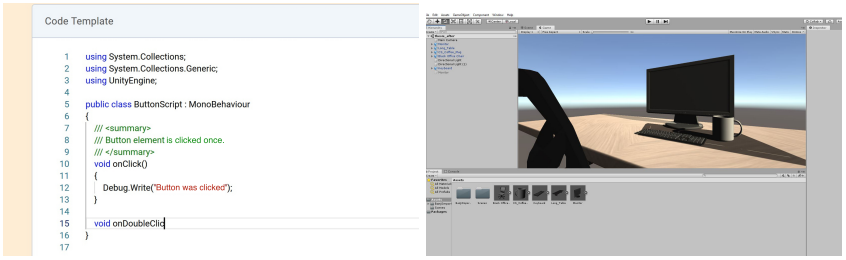


Fig. 7: Adding some button logic in the button script (left); Object composition in Unity with better quality (right).

4 Related Work

AR and VR applications are widely spread in various application domains (e.g., health [6], education [8], etc.). As described in previous work (e.g., [7], [3]), the prototyping and development of AR and VR applications is a cumbersome and challenging task. Multiple research projects and commercial tools are trying to

make AR/VR application prototyping and development easier, quicker and overall more accessible. One can roughly categorize them into two types: repositories and authoring tools. Unity Asset Store[5] and Google Poly[6] are both repositories, albeit they serve different purposes. Poly is a sharing platform for digital art, mostly 3D models and its sole purpose is to enable designers to browse through the catalog and look for useful models. Unity Asset Store, on the other hand, distributes assets, whereby assets refer to whole components of programs, including art, audio and business logic. Furthermore, Unity Asset Store is neither limited to AR/VR assets nor does it specifically target them. In contrast, Poly is focused on AR/VR development, although it is not necessarily limited to these applications. So, object and asset repositories enable user to share and re-use their content over multiple projects. Nonetheless, they come with some pitfalls. If a repository is too restrictive, Poly for example, then its use is limited as it only allows plain 3D objects with no logic. Unity Asset Store contains almost every type of component that can be made in Unity Engine, which is why it is time-consuming to find an asset that fits your needs. Then, there are authoring tools, like DART [4] and ProtoAR [5], both of them targeting AR. They do not provide content in terms of 3D objects, but rather support the prototyping and/or development of applications. While DART is based on Macromedia Director and thus acts as a development environment, ProtoAR serves as a prototyping tool only. As it only simulates AR functionality, ProtoAR does not fully support development. DART assist prototyping with its ability to record sensor data and play them back outside of the targeted environment. In contrast to ProtoAR, DART stronger supports development. DART's approach, however, is inflexible, because it ties the development to a specific tool for the whole process, from design prototyping to actual implementation. This approach might have been useful in the early 2000s, because of limited AR framework, today, however, it is too restrictive. The approach of ProtoAR, a tool for prototyping only, seems more appropriate. Designers can use ProtoAR to create mock-ups for their UI design, while developers can work with a tool set of their own choice. While this is more flexible, it does mean the designers cannot extend their prototype gradually, as at one point they have to switch their tool set.

Therefore, our solution, combines both approaches. It is an object repository, providing AR / VR specific objects, that can be used in any development environment. These objects also contain templates for logic, thus assisting the developers when prototyping or developing. This can be combined with a tool set for designers to create digital mock-ups from paper and previewing their design on an actual AR / VR device. The repository can be accessed via REST API, therefore the supported IDEs and devices can be extended.

5 Conclusion and Future Work

In this demo paper, we presented our approach for object round-tripping which supports the rapid prototyping of AR/VR applications. Designers can add new

[5] `https://assetstore.unity.com/` [6] `https://poly.google.com/`

objects to the repository by using existing graphics or by creating new ones by taking pictures or via photogrammetry. Objects can be classified to support their reusability. Metadata, graphics, and business logic are stored separately, enabling easier and independent editing. Additionally, this enables storing each of those units to be stored either as a database for metadata or the file system for binary objects. With the case study, we have shown an exemplary use of the object repository and how the designers can benefit from the different provided features. While the concept is promising, several topics can be addressed by subsequent research. As the object repository is designed with extensibility in mind, using a platform-independent REST-API for all central functionalities, the approach can be extended to a variety of client devices. For example, a dedicated photogrammetry scanner or AR devices, like Microsoft's HoloLens can be added. They may increase the object quality without increasing the workload significantly. Additionally, the repository can be extended by object recognition to enable automatic tagging of objects.

Acknowledgements We would like to thank our student Niklas Junge who helped implementing the presented approach.

References

1. Foster, S., Halbstein, D.: Integrating 3D Modeling, Photogrammetry and Design. SpringerBriefs in Computer Science, Springer London (2014)
2. Gandy, M., MacIntyre, B.: Designer's augmented reality toolkit, ten years later: Implications for new media authoring tools. In: Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology. pp. 627–636. UIST '14, ACM (2014)
3. Krings, S., Yigitbas, E., Jovanovikj, I., Sauer, S., Engels, G.: Development framework for context-aware augmented reality applications. In: EICS '20: ACM SIGCHI Symposium on Engineering Interactive Computing Systems. pp. 9:1–9:6. ACM (2020)
4. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.D.: Dart: The designer's augmented reality toolkit, `https://ael.gatech.edu/dart/`
5. Nebeling, M., Nebeling, J., Yu, A., Rumble, R.: Protoar: Rapid physical-digital prototyping of mobile augmented reality applications. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. pp. 353:1–353:12. CHI '18, ACM (2018)
6. Yigitbas, E., Heindörfer, J., Engels, G.: A context-aware virtual reality first aid training application. In: Proceedings of Mensch und Computer 2019. pp. 885–888. GI / ACM (2019)
7. Yigitbas, E., Jovanovikj, I., Sauer, S., Engels, G.: On the development of context-aware augmented reality applications. In: Beyond Interactions - INTERACT 2019 IFIP TC 13 Workshops. LNCS, vol. 11930, pp. 107–120. Springer (2019)
8. Yigitbas, E., Tejedor, C.B., Engels, G.: Experiencing and programming the ENIAC in VR. In: Mensch und Computer 2020. pp. 505–506. ACM (2020)