A Shrinkwrap Method for Quickly Generating Virtual Prototypes for Extended Reality

A Shrinkwrap Method for Quickly Generating Virtual Prototypes for Extended Reality

Paolo Senesi[1], Barbara Lonzi[2], Alessandra Papetti[1], Michele Germani[1], and Marco Mandolini[1]

[1] Università Politecnica delle Marche, Ancona, Italy
m.mandolini@staff.univpm.it
[2] Benelli Armi S.p.A., Urbino, Italy

Abstract. Extended Reality (XR) applications require Photorealistic Virtual Pro- totypes (PVPs), usually obtained by generating a polygon-based textured model of the original CAD (Computer Aided Design). Getting PVPs requires specific ren- dering and texturing software tools that experienced technicians use. Automatic simplification and conversion approaches from CAD to XR exist but are almost all focused on models without textures. The paper aims to establish a semi-automatic method for creating low-poly PVPs for XR starting from 3D CAD models. The process, implemented in Blender, consists of several steps. First, starting from a high-poly model imported from a 3D mechanical CAD system, the modelling cage (a low-poly simplified version) is designed. Second, a low-poly variety of the CAD geometries are generated using Blender modifiers (i.e., shrinkwrap, subdivision surface and Boolean). Texture mapping is carried out on the cage. Then, by combin- ing Shrinkwrap, Boolean and Subdivision Surface modifiers, the cage is projected on the CAD-imported high-poly model. Thus, it becomes a low-poly version of the same geometry. Finally, the Normal Baking process adds high-frequency details (e.g., engravings). Thanks to the generation of a UV-Mapped cage wrapping up the component, in case of local modifications to the latter, Blender semi-automatically updates the PVP. The method was used on three stock versions of a sporting rifle. With an average duration of 23 min, the proposed approach more than halved the time required with manual modelling techniques.

Keywords: Virtual Prototyping Extended Reality Texture Mapping 3D Modelling

1 Introduction and Literature Review

Extended Reality (XR) is gaining increasing interest because it can reduce lead time and increase user involvement in product development. The seamless and automated integration of all product data defined in CAD (Computer Aided Design) systems toward XR systems is the primary enabling technology to support CAD-XR integration [1].

XR viewer applications require Photorealistic Virtual Prototypes (PVPs), usually obtained by generating a polygon-based textured model of the original CAD. Getting PVPs requires using specific rendering and texturing software tools. Nonetheless, the

experience of the technicians who, thanks to computer graphics skills, can carry out this transformation is fundamental. As this is an entirely manual process, each modification resulting from the 3D model implies a new, complex, and lengthy modelling and texturing process. In the case of customised products, where changes are very recurrent, this traditional approach, which would require the execution of all phases manually, is not sustainable.

Modelling, texturing and XR programming are the three main processes that comprise the development process [2]. The 3D model of the virtual prototype is initially created using 3D CAD modelling software. Then, the model is rendered using computer graphics techniques (e.g., texture mapping). The 3D CAD software tool can be used directly for this operation. The virtual prototype is then imported into the game engine to create the XR software and the associated graphic UV texture maps.

The conventional method for utilising a CAD model in an XR system is to mesh the original model and send it to a rendering graphics system [3]. This approach is ineffective for XR systems due to the enormous amount of triangles that were generated for the following reasons [4]:

- Meshes produced by tessellation for local viewing have a high triangular density and a high transmission burden.
- VR systems sometimes produce sluggish rendering and subpar 3D manipulation.

For these motivations, Tang and Gu suggested a novel method for CAD model conversion and simplification for a VR system [5]. Also, Harlan et al. [6] recently developed an XR-CAD platform to interface with a game engine and CAD software efficiently.

Other research aiming at simplifying CAD models for real-time rendering is available in the literature. Dunming et al. presented a GPU-based compression method for rendering massive aircraft CAD models [7]. Lorenz et al. produced an automated CAD to XR conversion workflow based on model complexity reduction, animation and kinematic adoption [1]. Previous simplification approaches are available in [8] and [9]. Despite the benefits obtained, the procedure does not handle textured models.

An interesting approach to converting CAD models (considering textures) for real-time rendering is given in [10]. The authors used the DATASMITH plugin of Unreal Engine to overcome the problem of UV mapping [11] complex polygonal models exported in STL format. Visual anomalies, such as stretched or blurred textures on the edges, might occur without any processing. Unfortunately, the cited paper does not explain how the method can be employed to quickly manages CAD model revisions.

The paper aims to establish a semi-automatic method for creating low-poly PVPs (to be used in XR applications) starting from virtual models generated by 3D CAD systems. The process, implemented in Blender, consists of several steps. First, starting from a low-poly simplified version (modelling cage) of the CAD-imported high poly, obtained once by 3D modelling, it is possible to replicate a low-poly variety of the CAD geometry using Blender modifiers. Second, the texture mapping is carried out on the cage. These two activities are carried out once for each product. Then, by combining Shrinkwrap, Boolean and Subdivision Surface modifiers, the cage is projected on the CAD-imported high-poly model. Thus, it becomes a low-poly version of the same external geometry. Finally, the Normal Baking process adds high-frequency details (e.g., engravings).

Thanks to generating a UV-Mapped cage wrapping up the component, Blender can quickly update the virtual model semi-automatically in case of local modifications to the latter. In this way, the manual contribution of a technician is minimal and limited to modelling and texturing the wrapping cage.

2 Shrinkwrap Methodology

2.1 Summary

The method was developed in Blender and is based on modifiers, which are automatic algorithms for non-destructive operations on polygonal objects (meshes), including the possibility to combine, subtract, or mutually influence meshes with each other.

The method works by wrapping a simplified and UV-mapped mesh around the 3D model of the component imported from CAD. The mesh, called *Shrinkwrap Cage*, is morphed to the external geometry of the CAD-imported high poly without high-frequency detail. The output of the process is a lightweight low-poly model to use in combination with a normal map, exported as a 16-bit raster image that must be applied to the material to reproduce the high-frequency detail on the external surface of the model. The *Shrinkwrap Cage* is required to follow the overall shape of the original component's external geometry. It must be manually modelled, paying attention to how the modifiers act on it so that the correct behaviour of all modifiers is ensured. The seams for UV mapping must be marked on it to shrinkwrap a textured mesh. The manual modelling operation needs to be performed only once for each component because the *Shrinkwrap Cage* can be reused for subsequent updates of the prototype design. The duration necessary to obtain the *Shrinkwrap Cage* for an individual part is comparable in magnitude to the time required for manual 3D prototyping. However, since the operation is not repeated in the update cycle, this time will not be considered in the time process.

The operation sequence is described in detail in Fig. 1, subdivided into three macro-steps. The activities in the **Macro Step 1** include importing, adjusting the elements to align the imported model and *Shrinkwrap Cage* until the desired effects of modifiers on the cage are obtained, and creating copies that will be used for the Normal Baking process, which constitutes the **Macro Step 2** on its own. Normal Baking is the automatic process available in Blender that allows for creating detailed normal maps from the external surface high-poly models. The operations in the **Macro Step 3** involve post-processing the obtained detailed normal map and exporting the model as an FBX in combination with its corresponding normal map.

Then, the files must be imported into the graphics engine within the project folder of the XR application project. Both activities in the first and third macro steps are primarily manual-like operations. At the same time, the Normal Baking process is fully automated.

2.2 Modifiers and Cutting Tools

As shown in Fig. 2, the Shrinkwrap modifier allows for reproducing the imported model's low-frequency external features. Still, it does not work correctly where features such as grooves, holes or sharp edges are located. Therefore, it is necessary to combine the Shrinkwrap modifier with other ones, in particular:

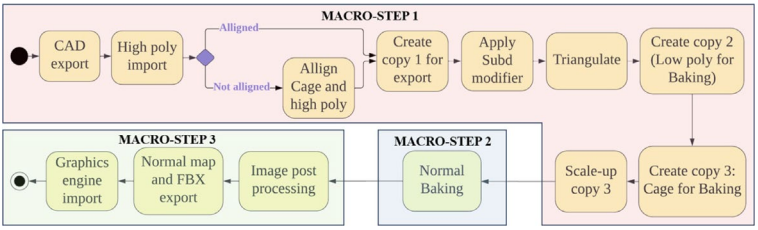

Fig. 1. Activity Diagram showing the operation sequence of the method.

- **Subdivision Surface modifier** (called "Subd" in Fig. 1): it increases the resolution of the *Shrinkwrap Cage*.
- **Boolean modifier**: it refines mid-frequency details by subtracting geometry.

The Boolean modifier is critical for mid-frequency features because it locally redefines the cage to have the same surface geometry as the 3D-modelled cutting object. In this way, the cage can adhere to the high-poly geometry, even in mid-frequency detail areas, through the effect of the Shrinkwrap modifier (Fig. 3).

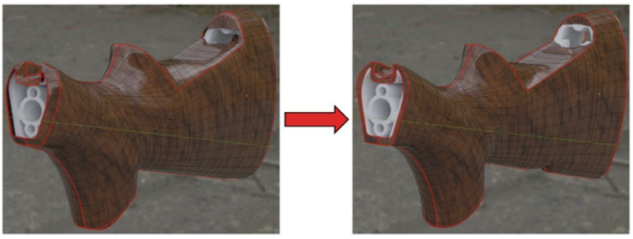

Fig.2. Effect of the Shrinkwrap modifier acting on the UV-mapped cage, with the high-poly model selected as the target object.

The cutting objects are polygonal objects previously modelled in combination with the Shrinkwrap Cage, so they are ready to use in the BLEND file relative to the component, in a correct relative position to the modelled cage. Through Boolean modifiers,

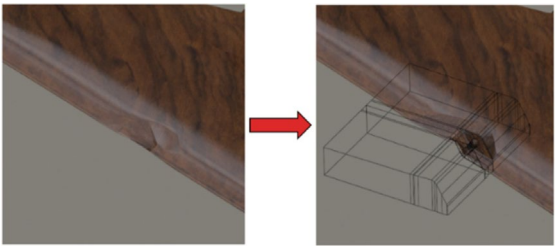

Fig. 3. Improvement of the effect of the Shrinkwrap modifier thanks to the Boolean modifier: the Cutter, displayed as a wireframe, locally refines the cage so that the mid-frequency feature is correctly wrapped.

the Cutters subtract geometry from the *Shrinkwrap Cage* and locally redefine its shape and topology before wrapping to adhere well to the high poly model, even in the case of sharp geometries.

2.3 Baking Normals

In Blender, Normal Baking is carried out by three actors:

- High-poly model: its external surface details are projected on the normal map.
- Low-poly model: the model on which the normal map creation is based.
- Baking Cage: it defines the volumetric boundary of the Baking process.

The Baking Cage is obtained by copying the low-poly model and scaling along the normals. Moving all faces along their external normal direction makes it possible to have a uniform offset between all faces of the different meshes. By doing so, the Baking Cage envelops both the high-poly and low-poly models. Based on the UV-mapping of the low-poly model, a normal map is produced through the Baking process. It reproduces the desired surface details if appropriately combined with the object material. The obtained normal map often needs to be cleaned using image editing software to avoid noisy or unwanted surface details data.

3 Validation

3.1 Case Study

The developed method was validated on the stock component of a sporting rifle to test the applicability and flexibility of the procedure. The model updates were simulated by switching between three high-poly stock versions and testing all possible combinations of old and new versions. The time required for each operation was measured.

Two cutters were modelled for geometric subtraction using the Boolean modifier for this class of component: (i) cutter for the lower sling hole feature and (ii) cutter for the anterior slot for the safety block.

The Normal Baking process was used to obtain the detail of the groove on the grip and the ridge near the front area of the comb. The three variants, compared in Fig. 4, used for the simulation are referred to:

1. “Base”: standard component version, used as a reference.
2. “Ver01”: version with an elongated rear section of the stock.
3. “Ver02”: version with slightly different geometry of the grip.

The processing time to update the virtual model version from one to another of the high-poly mentioned above models from the CAD environment was measured for six old-new combinations. To this end, three BLEND files were set up with their respective starting high poly. Following the procedure described by the activity diagram in Fig. 1, the time of all operations was measured.

Three tests were carried out for each of the six combinations, for 18 trials in total.

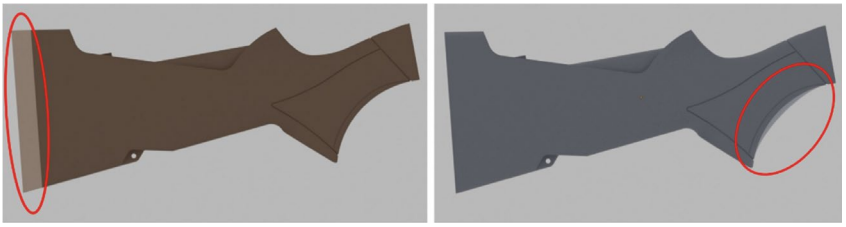

Fig. 4. Difference in geometry between Base and alternative versions used for the test: lightest areas represent the different features compared to the Base version. Ver01 on the left, Ver02 on the right.

3.2 Results and Discussion

Table 1 shows the execution time of the entire update process, subdivided into three macro steps as presented in Fig. 1, where the last row is calculated as the sum of the averages of the duration times for the macro steps, which were obtained from the three tests conducted for each macro actions. As stated in Sect. 2.1, the time to get the Shrinkwrap Cage was not included in the time process since it must not be repeated in each update cycle.

It can be noticed that there are no consistent variations in the execution time of the individual macro steps from one model to another and even less for the tests related to the same old-new model update combination. For this test, the same *Shrinkwrap Cage* was used for all pairings in each testing stage, requiring only a few minor manual adjustments before proceeding to the Baking step. This solution demonstrates certain operational flexibility for the case at hand.

Given that the estimated time to obtain the model using manual modelling techniques ranges from 1 to 2 h, it can be concluded that, with an average time of 23.4 min based on the tests conducted, the time required for updating the model has been reduced by more than half. Moreover, the developed methodology allows for repeatability as it is a standard procedure replicable by any design engineer, regardless of prior software experience, requiring the same learning curve as any other repetitive task. Regarding the quality of the models, it is essential to emphasise that the surface portions refined using the Boolean modifier may sometimes have some shading defects. Such imperfections are tolerable as they represent only a tiny part of the geometric extent of the model. Moreover, they are in areas that are not very visible.

Table 1. Model update process time for every combination and test number, reporting the average time across the three tests for each variety. Time expressed in minutes.

	Test n	Base→Ver01	Base→Ver02	Ver01→Base	Ver01→Ver02	Ver02 →Base	Ver02 →Ver01
Macro Step 1	Test 1	10	9	10	11	9	11
	Test 2	8	9	9	9	9	9
	Test 3	8	8	8	9	8	8

(*continued*)

Table 1. (*continued*)

	Test n	Base→Ver01	Base→Ver02	Ver01→Base	Ver01→Ver02	Ver02 →Base	Ver02 →Ver01
Macro Step 2	Test 1	10	9	9	9	9	10
	Test 2	10	9	9	9	9	10
	Test 3	10	9	9	9	9	10
Macro Step 3	Test 1	6	5	5	5	5	5
	Test 2	5	5	5	5	5	5
	Test 3	5	5	5	5	5	5
$T_{m,TOT}$		**24.0**	**22.7**	**23.0**	**23.7**	**22.7**	**24.3**

4 Conclusions

The paper presented a semi-automatic method for converting 3D CAD models into photorealistic virtual prototypes for XR applications. The approach was implemented in Blender and tested on the stock of a sporting rifle. The case study evaluated the time of generating new prototypes after revising a reference CAD model. Besides, also the quality of the renders was assessed. Despite the limited number of tests conducted, it can be stated that the update time of the model from CAD to the graphics engine for XR is less than half an hour. It comes to reducing more than half the time required with manual modelling techniques.

Future work aims to automate the entire process for real-time CAD-to-XR prototyping fully. Further research may employ Deep Learning to check the quality of the output 3D models and normal maps. Such a technology can be used to repair imperfections and defects automatically.

Acknowledgement. This work was supported by the 2014/2020 ERDF Regional Operational Programme of the Marche Region in the context of the research project 4USER – User and product development: from the virtual experience to the Regeneration of the model (CUP: B36G20001210007).

References

1. Lorenz, M., Spranger, M., Riedel, T., Pürzel, F., Wittstock, V., Klimant, P.: CAD to VR – a methodology for the automated conversion of kinematic CAD models to virtual reality. Procedia CIRP **41**, 358–363 (2016). https://doi.org/10.1016/j.procir.2015.12.115
2. Branislav, S., Dragan, C.: Mixed Reality and Three-Dimensional Computer Graphics. IntechOpen, London (2020). https://doi.org/10.5772/intechopen.77405
3. Jezernik, A., Hren, G.: A solution to integrate computer-aided design (CAD) and virtual reality (VR) databases in design and manufacturing processes. Int. J. Adv. Manuf. Technol. **22**, 768–774 (2003). https://doi.org/10.1007/s00170-003-1604-3
4. Qiu, Z.M., et al.: Geometric model simplification for distributed CAD. Comput. Aided Des.. Aided Des. **36**, 809–819 (2004). https://doi.org/10.1016/j.cad.2003.09.007

5. Tang, Y., Gu, H.: CAD Model's simplification and conversion for virtual reality. In: 2010 Third International Conference on Information and Computing, pp. 265–268. IEEE (2010). https://doi.org/10.1109/ICIC.2010.338
6. Harlan, J., Schleich, B., Wartzack, S.: Linking a game-engine with CAD-software to create a flexible platform for researching extended reality interfaces for the industrial design process. In: Proceedings of the 31st Symposium Design for X (DFX2020), pp. 169–178. The Design Society (2020). https://doi.org/10.35199/dfx2020.18
7. Dunming, T., Gang, Z., Lu, Y.: GPU based compression and rendering of massive aircraft CAD models. In: 2012 International Conference on Virtual Reality and Visualization, pp. 50–55. IEEE (2012). https://doi.org/10.1109/ICVRV.2012.8
8. Graf, H., Brunetti, G., Stork, A.: A methodology supporting the preparation of 3D-CAD data for design reviews in VR. In: Proceedings of the 7th International Design Conference. 1–2, pp. 489–495 (2002)
9. Stelzer, R.H.: Virtual reality based engineering collaboration as part of the product lifecycle management. In: Volume 3: 30th Computers and Information in Engineering Conference, Parts A and B, pp. 1327–1334. ASMEDC (2010). https://doi.org/10.1115/DETC2010-28250
10. Prada, E., Kolár, A.: Possibilities of convert cad models for real time rendering software. Tech. Sci. Technol. **3**, 220–228 (2020). https://doi.org/10.25140/2411-5363-2020-3(21)-220-228
11. Wikipedia the free Encyclopedia: UV mapping. https://en.wikipedia.org/wiki/UV_mapping. Accessed 23 May 2023