Linking a game-engine with CAD-software to create a flexible platform for researching extended reality interfaces for the industrial design process

Jakob Harlan[1,*], Benjamin Schleich[1], Sandro Wartzack[1]

[1] Engineering Design (KTmfk), Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

** Korrespondierender Autor:*
Jakob Harlan
Lehrstuhl für Konstruktionstechnik KTmfk
Martensstraße 9
91058 Erlangen
Telefon: +49 (0)9131/85-23659
Mail: harlan@mfk.fau.de

Abstract

Driven by the need to develop highly complex products in short development cycles, many companies are aiming at fully digitized design workflows and continuous data and information flow. XR-interfaces can extend current CAD-functionality to allow the usage of CAD-data in more stages of the product design process. In this paper, a platform linking a game-engine and CAD-software to efficiently explore XR-CAD Interfaces is presented. An architecture meeting the requirements of research environments is designed and implemented using the unreal engine and Siemens NX. The proposed system allows to efficiently explore interfaces for design sketch creation using different hardware setups.

Keywords

Virtual Reality, Extended Reality, Computer-Aided-Design, Natural-User-Interface

1. Motivation

Driven by the need to develop highly complex products in short development cycles, many companies are aiming at fully digitized design workflows and continuous data and information flow. This is because such digital design workflows offer advantages, such as the minimized need for data transformation and increased data traceability. However, current authoring tools and computer-aided design (CAD) software are not practical for all stages of the product development process. Particularly in the early phases, where different designs are explored and the detailed shapes are still in discussion, the precise and concrete nature of CAD software hinders progress. For this reason, product designers rely on more rapid and flexible tools like hand drawings. However, extended reality (XR) interfaces, which are all interfaces using virtual, mixed, or augmented reality, can extend the functionalities of CAD software. Advantages of XR-CAD interfaces can range from easier access for inexperienced users, faster interactions, improved size perception to simply a more compelling experience [1–3].

While the idea of combining XR and CAD is not new [1], to this day CAD-software makes little use of XR's capabilities. Tough most of the commercial products have a virtual reality (VR) visualization mode, none use the added in- and output possibilities of modern XR hardware to their full potential. Especially the creation and manipulation of geometric data is not supported well.

However, the development of new XR-CAD interfaces is a complex task. Researchers have to understand the needs of product developers and have enough knowledge of human-computer-interaction and XR computer graphics to implement and evaluate their approaches. To overcome these challenges, modern game engines are the perfect starting point to develop a platform for exploring XR-CAD interfaces as they encapsulate many useful features. They provide state-of-the-art rendering technology, support a large range of in- and output devices, and interactive editors. Widespread engines additionally have great amounts of resources for learning and problem solving available. Motivated by this, this paper presents a flexible platform, combining a CAD-software with a game engine, for researching XR-CAD interfaces and its usage in a current research application (see Figure 1).

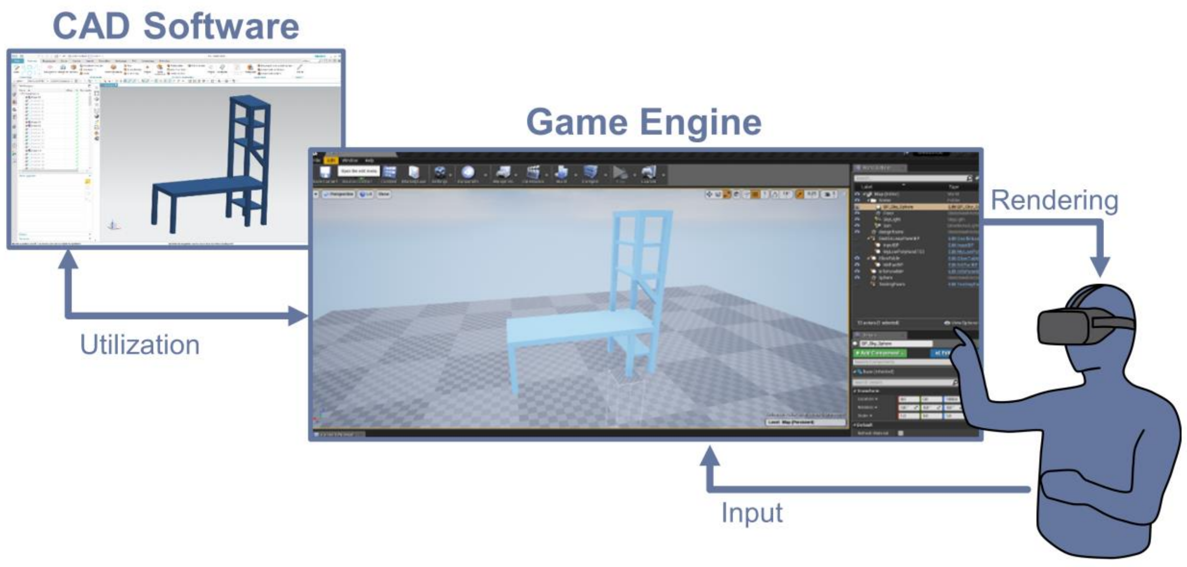

Figure 1: Overview of the platform connecting a CAD System with a Game Engine

2. Related Work and Research Problem

This section provides an overview of related scientific work and is split into three parts. In subsection, 2.1 works towards the integration of VR and CAD are presented. Special attention is given to the used technical implementation. The following subsection 2.2 describes in which ways other research fields have used game engines to their advantage. The last but one subsection 2.3 summarizes some recent publications using combinations of CAD software and game engines for different tasks in product design, while the last subsection highlights the research problem and target.

2.1. CAD in XR

One of the first publications bringing VR and CAD together is from 1999 when Julien Berta presented *Dassault System's* approach for integrating VR and CAD in [1]. He compares both technologies to reveal their differences. While both use 3D geometry data, they employ different formats for their specific use cases. Additionally, CAD engines were not capable of real-time interactions, which is essential for VR. Also, he identifies the difference between the used in- and output. CAD relied on a classic mouse, keyboard and monitor interface, VR offered stereoscopic displays with gloves and motion trackers for user input. He shows how these differences were overcome by integrating VR technologies into the core of the Catia CAD software.

Patrick Bourdot and his team have contributed immensely to the CAD in VR research. In 2010 they presented the *VRAD* (Virtual Reality Aided Design) prototype [3]. It was based on the *OpenCASCADE* geometric kernel and was later adapted to use *CATIA* [4]. The system utilized the device management and visualization system *EVI3d*, which was developed in the same research facility earlier. They were able to use a naming technique to connect the B-Rep data to the parameterized CAD features. Based on this they developed interactions directly changing the underlining geometric parameters.

In recent years driven by the rise of affordable consumer VR hardware, a lot more research on how VR can be used in product development has been conducted. For example, Fechter et al. worked on natural-user-interfaces for interacting with CAD data. They used consumer-grade visual body-, hand- and finger-tracking as input-devices to research intuitive interfaces for different product development tasks. They created a custom framework combining *Leap Motion* SDK, *Oculus* SDK, *Nvidia PhysiX*, *Open Scene Graph*, and *Ansys SpaceClaim* to prototype and evaluate their developed interactions [5]. First assembly modeling was implemented [2] and recently design sketch creation was explored [6].

All these presented developments used custom implementations for the VR interaction frameworks. By using the specific APIs, this approach will give the researchers the most freedom. However, changing the hardware setup or the used CAD-system will require substantial implementation effort. Additionally, new contributors will need a long familiarization period.

2.2. Game engine usage in research

Using game engines for research is not a novel idea. In 2002, the *Communications of the ACM* published a special issue motivating the usage of game engines for scientific research [7]. While the practical tips on choosing and licensing an engine are outdated, the principle still holds. A game engine can provide state-of-the-art 3D rendering and interaction features with little effort. An early example is the research of Shiratuddin and Thabet, who used a precursor of the *unreal engine* for virtual office building walkthroughs [8].

As game engines developed and became more flexible, more researchers used their potential. For instance, in 2007 Carpin et al. presented a robot simulator based on the *unreal* *engine 2.0* [9], and in 2009 Indraprastha and Shinozaki investigated using *unity3D* in urban design studies [10].

Today, two engines dominate the market for both game and general-purpose applications: *Unity3D* [11] and *Unreal Engine 4.0* [12]. Lv et al. shared their experience with rebuilding a specialized tool for molecular visualization with *Unity3D* [13]. They remarked the good performance, ease of development, the large number of target platforms, and community-driven support.

2.3. CAD and Game Engines

In contrast to other scientific fields, research in product development has just recently started to use the benefits of game engines.

This year, Pereira and Ellman presented a framework to transform a CAD model into a physics-based prototype simulated in Unity3D [14]. By using the software from *AGX Dynamics* to both define the physics in *SpaceClaim* and simulate the model in *Unity3D*, they were able to transform a harvester CAD Model in just two workdays into a simulated model with VR interactions.

Ekströmer et al. used the *Unreal Engine's* extensive real-time lighting features to support automotive designers to develop car lamps [15]. Exterior and interior light designs were developed and evaluated with and without the usage of VR. By utilizing an expert evaluation they concluded that game engines can help with design ideation in the early stages of the design process.

Feeman et al. developed a VR-CAD modeling application by connecting *Autodesk Inventor* and the now discontinued *Autodesk Stringray* game-engine via text-based network communication [16]. They studied how the modeling process of simple objects and the resulting models diverge between a classic CAD environment and the developed VR-CAD modeling application.

2.4. Research problem and target

The efficient exploration of novel XR-CAD interfaces requires a platform that is capable of performing CAD operations as well as supports quick, accessible development for the whole range of XR devices. However, there is no tool meeting these requirements singlehandedly.

As mentioned in section 2.1, using a custom framework connecting the devices' and program's APIs will provide maximum flexibility at the cost of implementation effort and code complexity. That complexity will lead to long familiarization periods for new students and researchers. Motivated by these shortcomings and the growing interest in coupling game engines with CAD software as highlighted in section 2.3, this work tries to answer the question: How should a research platform supporting a wide range of XR hardware and CAD tasks be designed? In this context, the aim is not the development of a production-ready application, but to provide a platform enabling quick iterations for researching novel XR interactions, which is also usable for students and researchers.

3. Used methods and approaches

Combining CAD-software with a game engine brings together all features necessary for a XR-CAD Platform. The CAD-tool provides geometric data management as well as all necessary operations to modify the data. It also enables loading and saving widespread CAD data formats. The game engine comes with everything needed for XR Interface development. This being extensive rendering capabilities, a wide spectrum of supported input and output devices, an editor for interactive development, and many resources for user support. While designing the software architecture the aim was to keep the system's core as independent from the environment as possible. This eases changing the setup without major revisions. Therefore, both the communications to the CAD-software and the input handling are handled by separate modules. Without the loss of generality, for this work, an *unreal engine* [12] application that accesses the CAD functionality of *Siemens NX* [17] via the *NXOpen c++ API* [18] was created. While these choices are the best fitting for us, an adaptation of the developed architecture to other game engines and CAD-systems should be possible.

4. Software Architecture

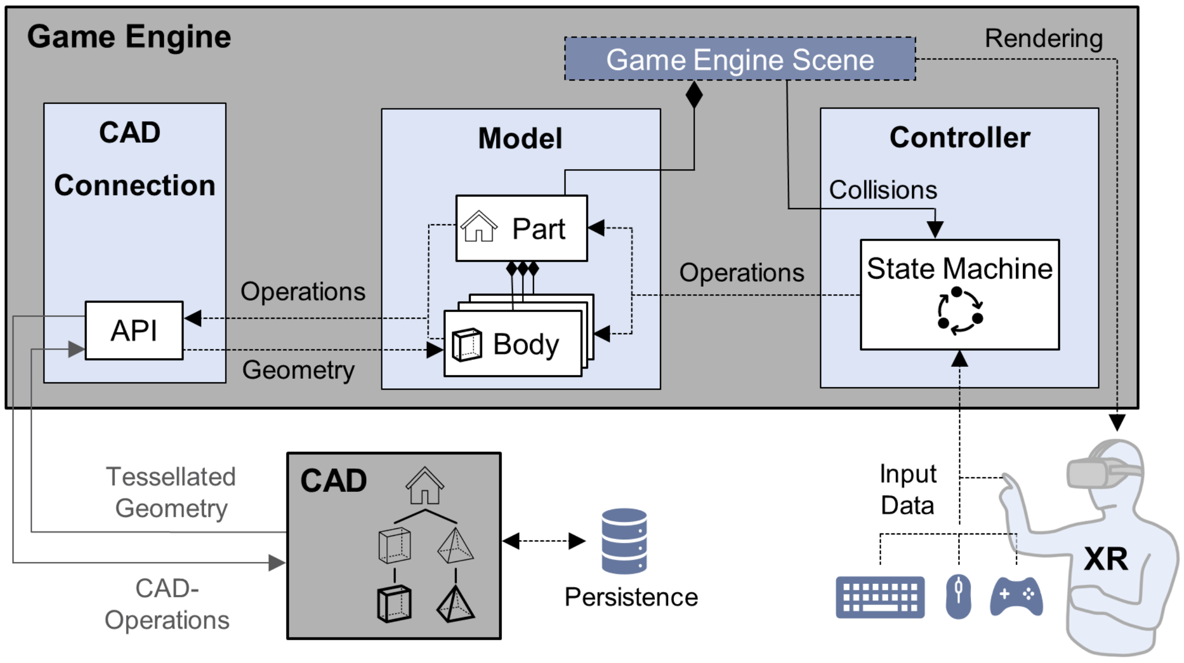

Figure 2: Software architecture with the three main modules: CAD Connection, Model, and Controller.

The platform is a game engine project and implemented using mostly *c++*. The *unreal engine* provides a visual scripting system called *Blueprints* which is also used. As mentioned in section 3, the project is split into three major modules: CAD connection, Model, and Controller. The responsibilities and functions of each will be summarized in the following subsections. While the CAD connection is written only in *c++*, the model and controller classes have *Blueprint* representations. This allows users to use large parts of the platform without knowledge of the complex programming language *c++*.

4.1. Model

The central module of the platform, *model*, is responsible for the geometric data and located in the center of Figure 2. The two main classes, *Part* and *Body*, are the engine-site representation of the CAD geometry. Both are derived from build-in engine classes and inherit many useful features.

The *Part* class describes the whole geometric model and uses the engines actor class as its parent. As an actor, the *Part* can be placed in the engine's scene with a transformation allowing it to be translated, rotated, and scaled. A *Part* can contain any number of instances of the *Body* Class. The *Body* class is derived from an actor component class. By being a scene component, it can be attached to an actor in the engine's scene and has a relative transform to that actor. The specific class it is derived from is called "ProceduralMeshComponent" which allows loading and reloading triangle mesh data at runtime, which is necessary to update the

CAD geometry data. This composition of bodies and parts allows the engine to automatically render the geometry data as well as provide collision and picking capabilities for the models.

This setup would be possible in other game engines as well. For example, in *Unity 3D* [11], the part would be a *GameObject* and the body a *MeshComponent*.

Besides representing the geometry, the model module also provides the operations to modify that data. In the current state of the platform methods of constructive solid geometry are implemented. These include the addition of primitive bodies such as blocks, spheres, and cylinders, removal of a body, moving a body, copying a body, and combining bodies through boolean operations. Also, some operations not performing directly on the geometry like undo, loading, and storing part files are available. Most operations concern the underlying cad data and those are issued to the *CAD Connection* module which is described in the next section.

4.2. CAD Connection

Shown on the left of Figure 2 is the *CAD Connection* module. This module is responsible for all communications to the CAD software. This allows for all API specific code to be in one place and would ease changing the used CAD program. The module starts and holds the CAD program session without opening a window. By being modeled as a singleton, this guarantees that always exactly one CAD session is active and can be used.

When the *model* module issues operations to the *CAD Connection*, it internally translates the operations and parameters appropriately for the CAD API. Afterward, new or changed bodies are reloaded. To load a body's geometry into the engine, first, the CAD software's tessellation is used to create a triangle mesh from the parametric geometry. Next, the data is marshaled into the engine format and then a new body can be created or an existing one's geometry updated. For the operations to be performed on the correct bodies a unique identifier provided by the CAD software is used to connect the CAD-side and engine-side representations.

Additionally, the *CAD Connection* is responsible for the data persistence, meaning it issues loading and saving of the CAD files.

4.3. Controller

The last module is the controller, shown on the right of Figure 2. It collects all inputs and decides which operations are executed. Currently used inputs are the positional tracking data from head-mounted-displays, finger-respectively hand-tracking data, and keyboard strokes. Introducing further inputs such as handheld controllers would be easy, as the engine provides everything necessary.

Driven by the input data, the controller uses the in-engine collision mechanics to detect which object is interacted with. Currently, the collisions of the user's fingertips with the bodies and a menu are detected.

Using a state machine, the user input and the collision data is interpreted into actions. The state machine ensures that only intended operations are possible in every given moment. For example, while a body is grabbed and moved it should not be possible to simultaneously issue a deletion command. Operations acting on the CAD geometry are forwarded to the *model* module.

The controller and specifically the interaction state machine is the main region where new interactions are developed, given that the necessary CAD-operations are implemented. An exemplary state machine used for the creation of design sketches from primitive bodies can be found in Figure 4.

4.4. Call sequence

The sequence diagram in Figure 3 summarizes the call sequence and information flow in the developed platform. It shows the processes necessary when a Boolean operation is executed. Parts of this example are specific to the application presented in section 0. First, on the far right, the user's hand movement is picked up by an optical sensor. The *Controller* uses the input data to detect a tapping gesture and uses the engine's capabilities to detect which bodies the fingertip is colliding with. When the second body is tapped, the *Controller* issues a Boolean operation command on the *Part* object including the information which bodies and which Boolean operator were selected. The *Part* is now responsible to forward the operation to the *CAD Connection*. The *CAD Connection*, in turn, uses the bodies' identifiers to create the *API* calls. The CAD software executes the operation, updates its internal model, and tessellations of all changed and newly created bodies are acquired. These B-rep models are marshaled from the cad's to the engine's data format and are then used to create new bodies. If the operation was successful the new bodies are registered in the engine scene and the old ones deleted. This whole process is done within one frame and in the next frame, the user sees the new models rendered by the engine.

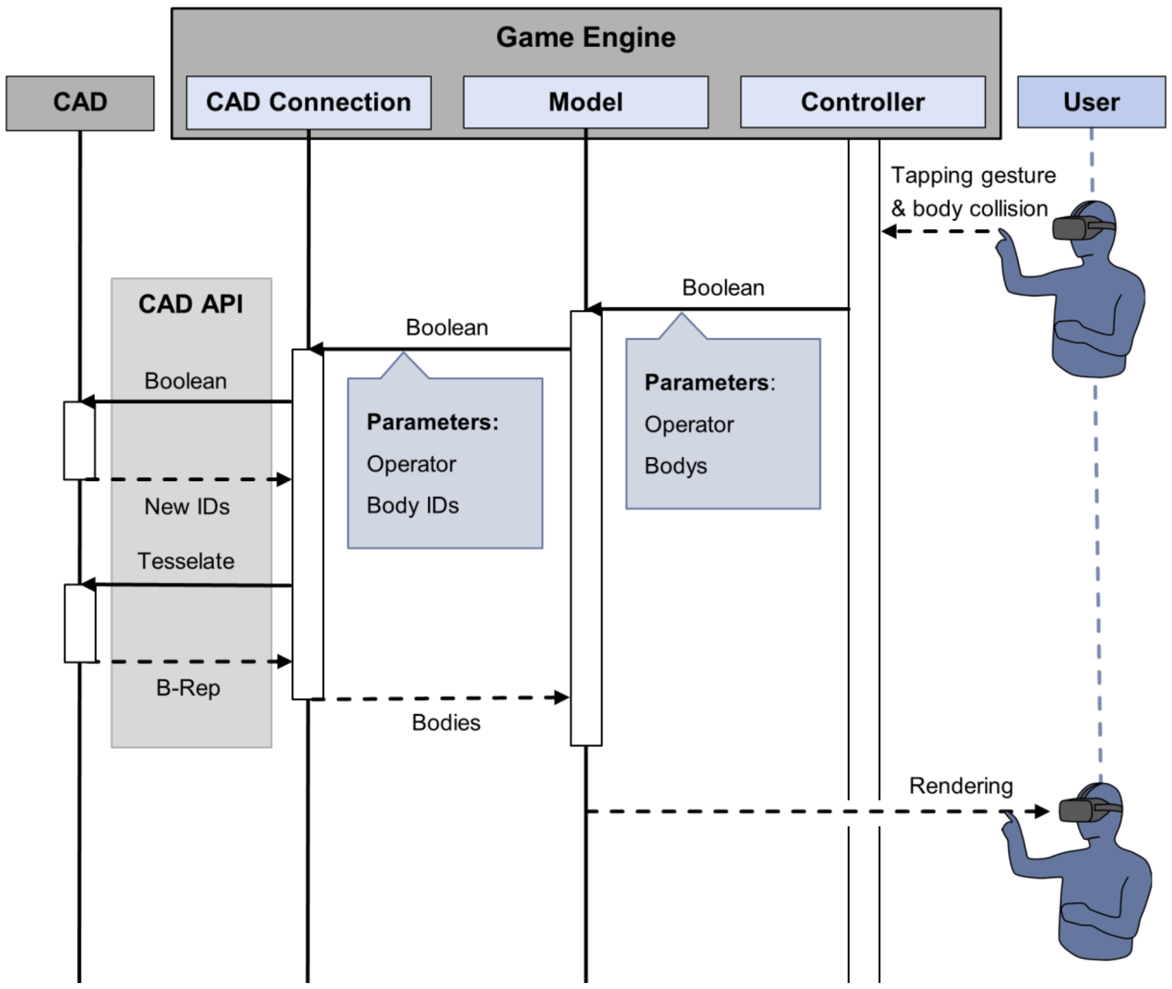

Figure 3: Exemplary sequence diagram showing the call sequence of a Boolean operation.

5. Application and Benefits in Research

The developed platform is currently used to research hand-tracking based natural-user-interfaces for design sketch creation in VR. The research's first goal is to find natural interactions best suitable for the task and later we aim to evaluate the developed interface against traditional design sketch methods like hand drawings.

The developed interface uses hand and finger gestures to create and modify primitive geometric building blocks. Cuboids, cylinders, and spheres with custom dimensions can be created with a simple gesture. The bodies can be grabbed for transformation and boolean operations allow further edits on the bodies, similar to constructive solid geometry. Screenshots of this application showing the possible operations are shown in the visualization of the state machine in Figure 4.

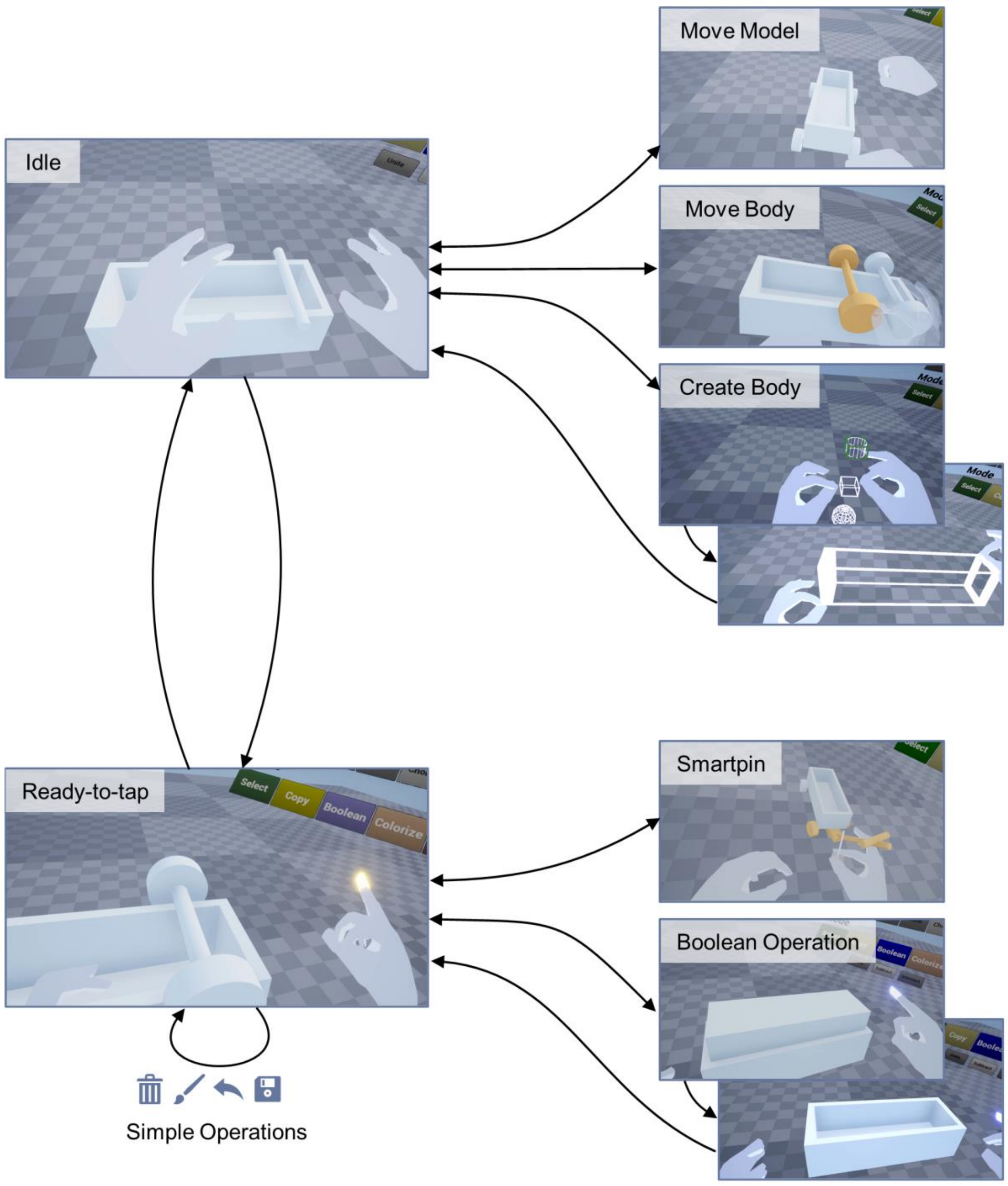

Figure 4: State machine in our research application: Natural interface for design sketch creation. The Smartpin interaction is used for precise movement [19].

The platform presented in this paper eases the development of these interactions significantly. It allows us to use different workstations with varying hardware. Currently, workstations equipped with the Oculus Rift, the HTC Vive Pro, and normal desktop monitors, all using a Leap Motion Sensor are in use. The vast amount of online resources helping with unreal engine development and the visual programming langue *blueprints* made it possible for students without computer-science background to contribute. The engine-based development should also help us to test the design sketch creation and develop new interactions on other display options, for example, an AR-Headset like the HoloLens 2 or a power-wall setup.

6. Discussion

In this paper, we have presented a flexible and extensible platform for developing XR-CAD interfaces by connecting a game engine to cad software. The application in our current research shows its suitability for developing interfaces for design sketch creation. We are convinced that extensions to the platform, following the proposed design principles, would allow it to be used for further research into applications of XR in other product-development tasks.

Possible extensions could include more inputs, outputs, sounds, physics, environments, and CAD features.

Danksagung

The authors thank the German Research Foundation for funding this research under grant number WA 2913/34-1.

Literaturverzeichnis

[1] BERTA, J.: Integrating VR and CAD. In: *IEEE Computer Graphics and Applications* Bd. 19 (1999), Nr. 5, S. 14–19

[2] FECHTER, M. ; WARTZACK, S.: Natural Finger Interaction for CAD Assembly Modeling. In: ASME (Hrsg.): *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 1: 37th Computers and Information in Engineering Conference*, 2017 — ISBN 978-0-7918-5811-0, S. V001T02A041

[3] BOURDOT, P. ; CONVARD, T. ; PICON, F. ; AMMI, M. ; TOURAINE, D. ; VÉZIEN, J.-M.: VR–CAD integration: Multimodal immersive interaction and advanced haptic paradigms for implicit edition of CAD models. In: *Computer-Aided Design* Bd. 42 (2010), Nr. 5, S. 445–461

[4] MARTIN, P. ; MASFRAND, S. ; OKUYA, Y. ; BOURDOT, P.: A VR-CAD Data Model for Immersive Design. In: DE PAOLIS, L. T. ; BOURDOT, P. ; MONGELLI, A. (Hrsg.): *Augmented Reality, Virtual Reality, and Computer Graphics*. Bd. 10324. Cham : Springer International Publishing, 2017 — ISBN 978-3-319-60921-8, S. 222–241

[5] FECHTER, M. ; WARTZACK, S.: Konzept für ein VR-System zur intuitiven Modellierung durch natürliche Interaktion. In: STELZER, R. (Hrsg.): *Beiträge zur virtuellen Produktentwicklung und Konstruktionstechnik* : TUDpress, 2016 — ISBN 978-3-95908-062-0, S. 561–570

[6] FECHTER, M. ; SCHLEICH, B. ; WARTZACK, S.: CAD-Gestaltmodellierung in VR für die frühe Entwurfsphase. In: *Konstruktion* Bd. 72 (2020), S. 69–74

[7] LEWIS, M. ; JACOBSON, J.: Introduction. In: *Communications of the ACM* Bd. 45 (2002), Nr. 1

[8] SHIRATUDDIN, M. F. ; THABET, W.: Virtual Office Walkthrough Using a 3D Game Engine. In: *International Journal of Design Computing* Bd. 4 (2002), S. 4

[9] CARPIN, S. ; LEWIS, M. ; WANG, J. ; BALAKIRSKY, S. ; SCRAPPER, C.: USARSim: a robot simulator for research and education. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, S. 1400–1405

[10] INDRAPRASTHA, A. ; SHINOZAKI, M.: The Investigation on Using Unity3D Game Engine in Urban Design Study. In: *Journal of ICT Research and Applications* Bd. 3 (2009), Nr. 1, S. 1-18–18

[11] UNITY: *Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations*. URL https://unity.com/. - abgerufen am 2020-06-18

[12] *Unreal Engine | The most powerful real-time 3D creation platform*. URL https://www.unrealengine.com/en-US/. - abgerufen am 2020-06-18. — Unreal Engine

[13] LV, Z. ; TEK, A. ; DA SILVA, F. ; EMPEREUR-MOT, C. ; CHAVENT, M. ; BAADEN, M.: Game On, Science - How Video Game Technology May Help Biologists Tackle Visualization Challenges. In: *PLoS ONE* Bd. 8 (2013), Nr. 3

[14] PEREIRA, J. G. ; ELLMAN, A.: FROM CAD TO PHYSICS-BASED DIGITAL TWIN: FRAMEWORK FOR REAL-TIME SIMULATION OF VIRTUAL PROTOTYPES. In: *Proceedings of the Design Society: DESIGN Conference* Bd. 1 (2020), S. 335–344

[15] EKSTRÖMER, P. ; WEVER, R. ; ANDERSSON, P. ; JÖNSSON, J.: Shedding Light on Game Engines and Virtual Reality for Design Ideation. In: *Proceedings of the Design Society: International Conference on Engineering Design* Bd. 1 (2019), Nr. 1, S. 2003–2010

[16] FEEMAN, S. M. ; WRIGHT, L. B. ; SALMON, J. L.: Exploration and evaluation of CAD modeling in virtual reality. In: *Computer-Aided Design and Applications* Bd. 15 (2018), Nr. 6, S. 892–904

[17] *NX*. URL https://www.plm.automation.siemens.com/global/en/products/nx/. - abgerufen am 2020-06-18. — Siemens Digital Industries Software

[18] *NX Open C++ Reference Guide: Main Page*. URL https://docs.plm.automation.siemens.com/data_services/resources/nx/10/nx_api/en_US/custom/open_c++_ref/index.html. - abgerufen am 2020-06-18

[19] CAPUTO, F. M. ; EMPORIO, M. ; GIACHETTI, A.: The Smart Pin: A Novel Object Manipulation Technique for Immersive Virtual Environments. In: *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, *VRST '17*. New York, NY, USA : ACM, 2017 — ISBN 978-1-4503-5548-3, S. 89:1–89:2