A Template Framework CRUD for Constructing Virtual Assembly Prototypes Supporting Multipath and Its Validation

Weichao Lin*
South China University of Technology

Liang Chen†
South China University of Technology

Zesheng Lin‡
Guangzhou Industry & Trade Technician College

ABSTRACT

The use of virtual reality technology for assembly training can provide an interactive and intuitive learning environment, enhancing learners' practical skills. However, the current methods for creating virtual assembly training content have limitations. Animation-based content requires manual arrangement of animation sequences by experienced users, which is inefficient. Interactive content based on the ASP algorithm incurs high computational costs and relies heavily on CAD data, imposing strict requirements on the model data. Furthermore, user interaction is constrained to the optimal solution calculated by the ASP algorithm, limiting the range of possibilities. To address these issues, we propose a framework called CRUD for the rapid construction of virtual assembly prototypes. This framework supports multipath assembly, enabling the simulation of real-world assembly scenarios. It can be implemented using common game engines, allowing end users to create virtual assembly sequences through simple configurations. We demonstrate the effectiveness and universality of the framework using a complex mechanical assembly case. We argue that enabling multipath assembly support in virtual training represents a significant step in making virtual simulation more realistic and practical.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interactive systems and tools; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Graphical user interfaces; Computing methodologies—Modeling and simulation—Simulation evaluation; Software and its engineering—Software notations and tools—Development frameworks and environments—Application specific development environments

1 INTRODUCTION

Cognitive training in mechanical structures is a key focus and challenge in the field of mechanical engineering. Traditional offline classrooms, which rely on textbooks, exercise books, and presentation slides, often lack intuitive structural representation and interactive engagement with learners. As a result, they fail to captivate students' interest and lead to suboptimal learning outcomes. While physical training shows reasonable effectiveness, it is limited by the availability of school model resources and incurs significant maintenance costs, which hinder its widespread adoption. In recent years, virtual technology has gained increasing significance in the manufacturing industry. Particularly in employee education and production tasks, the use of virtual reality has increased in industrial applications, such as training and product demonstrations. Cognitive assistance systems have demonstrated a high potential for improving efficiency, which in turn continuously creates new demands for content creation tools. Virtual reality-based assembly training offers learners an intuitive and interactive learning environment, addressing the limitations of traditional learning methods. In addition to interactive assembly training, visualizing assembly tasks through animations is also considered acceptable.

The creation of animated demonstrations of assembly processes is a laborious and time-consuming task, often carried out manually by graphic artists or engineers. Existing 3D modeling tools can assist in this process, but even for simple animations, they still require users with expertise to invest time in editing animation sequences [10]. In many current workflows for producing training materials, human designers are still responsible for making design decisions regarding assembly processes.

On the other hand, the field of industrial robotics has seen extensive research in Assembly Sequence Planning (ASP), which is a crucial aspect for the efficient production of large products with numerous components. However, these methodologies rely on 3D CAD data and result in significant computational costs. Even for simple assemblies, the computation of the explosion direction alone can take several minutes [6]. Such intricate calculations may not be easily embraced by general training teams. Furthermore, while the computed assembly sequence may represent a local or global optimum, it is imperative to acknowledge that in practice, there are multiple potential assembly sequences. It is also noteworthy that even suboptimal solutions can still meet the final assembly requirements. Therefore, for training teams seeking high realism and desiring to explore various possibilities and enhance training resilience, it may be more favorable to consider methodologies that offer more than just an optimal solution [8].

In order to address these concerns, we present a framework for creating multipath assembly sequences based on real-world physics principles to efficiently develop virtual assembly prototypes. This approach can be integrated into 3D game engines such as Unity 3D and Unreal Engine, allowing end-users to easily construct virtual assembly experiences that strictly adhere to real-world physics rules through simple configuration. This enhances the realism of the learning process and improves students' learning outcomes. The template method is thoughtfully designed and logically structured, requiring minimal resources to achieve a high-fidelity prototype, thereby reducing development costs and increasing system reusability.

This paper provides a comprehensive description of the framework's design and principles, emphasizing its uniqueness in comparison to existing methods. For the specific implementation of the framework, this paper utilizes Unreal Engine as an example and presents a complex mechanical assembly case to validate the framework's effectiveness and generality.

This paper makes the following contributions:

(1) Introducing an innovative framework for constructing virtual training prototypes capable of multipath assembly.

(2) Implementing this framework using the Unreal Engine and creating a prototype for a virtual assembly experience.

(3) Conducting an experimental study using the framework to develop an assembly training system for a complex mechanical product, with the aim of validating the effectiveness and generality of the framework.

*e-mail: linwc1126@gmail.com
†e-mail: earchen@scut.edu.cn
‡e-mail: linzesheng@outlook.com

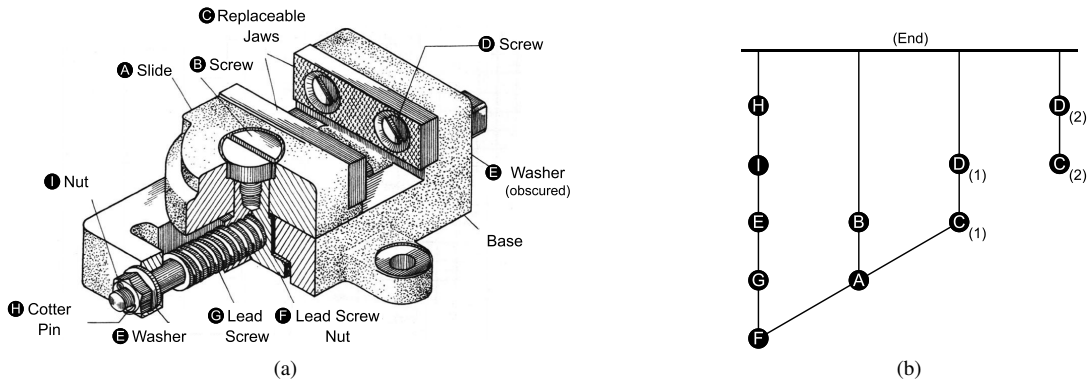


Figure 1: (a) The part composition and assembly structure of bench vise. (b) Hasse diagram is used to represent the relationship of the parts.

2 Related Work

2.1 Virtual Training in Mechanical Products

In the realm of mechanical products, assembly systems can be viewed as a flow process where components are systematically connected to obtain the desired final product [9]. The primary goal of integrating VR technology into education, particularly in the mechanical field, is to address the challenges associated with high risks, high costs, lack of intuitiveness, and inadequate hands-on experience in traditional practices.

Adas et al. proposed a learning environment based on virtual reality and augmented reality (VAR), which enables students to interact with virtual and real mechanical components in an interactive and guided manner [3]. Winkes et al. introduced a virtual reality workspace for assessment assembly decisions and training assembly operations [13]. Bharathi et al. compared VR devices with desktop solutions to investigate their effectiveness in learning how to perform product functional analysis, specifically in assembling components of a coffee machine [7]. Al-Ahmari et al. developed a virtual manufacturing assembly simulation system that provides an interactive workstation for evaluating assembly decisions and training assembly operations. The system provides feedback through multiple channels, including visual, auditory, and tactile sensations [5].

2.2 Assembly Sequence Generation

The rational design of assembly sequences is crucial for the assembly process, emphasizing the feasibility and rationality of assembly operations. For virtual training, recent research has focused on methods such as generating visual animations using game engines, conducting physical simulations in virtual training environments, and designing assembly instructions effectively.

Bahubalendruni et al. proposed a framework that takes parts, their assembly positions, and assembly sequence plans as inputs. Once the requirements of feasibility testing and interference testing are met, this framework can generate visual animations of assembly operations in a game engine [6]. Dyck et al. developed a virtual training environment to simulate the assemble of mechanical chucks, which described the precise motion sequence of each component in the assembly process. This high-fidelity assembly is achieved through the coordination of multiple colliders and constraints [8]. Agrawala et al. proposed design principles for creating effective assembly instructions and a system based on these principles. The system consists of a planner and a presenter responsible for determining the optimal assembly sequence and generating assembly demonstration graphics [4]. Gaarsdal et al. introduced an animation creation tool based on ASP that is capable of generating real-time exploded views of components for viewing in VR headsets [10]. Kalkofen et al. implemented an application that introduced explosion diagrams into augmented reality based on ASP [11].

Except for Dyck's training system, which allows for interactive training, the other systems can only provide demonstrations of the assembly process. Most current systems still rely on the initial involvement of professional engineers. This typically involves creating plans that include assembly direction vectors, movement distances, fixed sequences, and other parameters for each component. These plans are then used in preprocessing steps [6, 11]. Some systems cannot support assembly hierarchical structures with two or more levels [4]. Expanding the applicability of these systems to components would enable them to address larger and more complex use cases.

3 Solution

3.1 Design Process

Based on the preceding discourse, the current industry solutions for virtual assembly training can be broadly classified into two categories: one involves the manual arrangement of product assembly animations by designers or engineers, often utilizing software such as Maya or 3ds Max; the other is based on ASP algorithms for determining the most efficient assembly sequences. Both methods have their own limitations. The former necessitates the manual arrangement of animations for each moving component, which takes up a substantial amount of time and labor. Furthermore, this approach typically only showcases the most efficient assembly sequences. The latter heavily depends on CAD data, making it unsuitable for models with incomplete data and resulting in high computational costs. The outcomes are either globally optimal or locally optimal sequences, disregarding other potential sequences that may exist in reality.

Here, we describe our solution using a classic mechanical assembly case as an example. Consider a bench vise, which is often used as a typical example to describe assembly structures in engineering drawing courses. The marks $(A)-(I)$ in Figure 1(a) represent all the parts (excluding the base) that make up a bench vise. The collection of all parts is denoted as S. The complete assembly of this bench vise consists of several assemble tasks, where certain tasks can only be completed after others (e.g., screws on the jaw plate cannot be installed before the jaw plate is mounted). We can classify this as a partial order and represent the assembly relationships between parts as R [12]. The pair (S,R) forms a finite partial order set, and Figure 1(b) depicts its Hasse diagram representation.

To find a sequence for these assembly tasks, we can use a topological sorting algorithm, as shown in Algorithm 1, to obtain a compatible total order:

$$F \to G \to A \to E \to B \to C_1 \to D_1 \to I_2 \to C_2 \to D_2 \to H$$

It is important to note that this order represents only one of the

Algorithm 1 TopologicalSorting

Require:
$(S, \preccurlyeq)$ Finite poset.
Result: A compatible total ordering of S.

procedure TOPOLOGICALSORTING(S)
  $k \leftarrow 1$;
  while $S \neq \varnothing$ **do**
    $a_k \leftarrow$ a minimal element of S;
    $S \leftarrow S - \{a_k\}$;
    $k \leftarrow k + 1$;
  end while
return $[\, a_1, a_2, ..., a_n \,]$;
end procedure

potential sequences for these tasks. In reality, based on experimental observation, the most common assembly sequence is:

$$F \rightarrow G \rightarrow E \rightarrow I \rightarrow H \rightarrow A \rightarrow B \rightarrow C_1 \rightarrow D_1 \rightarrow C_2 \rightarrow D_2$$

The generation of multiple sequences is interpretable. With the continuous iteration of the topological sorting algorithm, the Hasse diagram may have non-unique minimal element, leading to different choices and subsequently different sequences. However, such sequence diversity is in line with real-world principles, and not every sequence necessarily adheres to the optimal sequence produced by ASP algorithms.

The reflexivity, antisymmetry, and transitivity of assembly relationships R among parts in set S demonstrate the diversity of assembly sequences. Transitivity reflects the relationships between parts in terms of their assembly order. In other words, if *part A* and *part B* are directly connected in the Hasse diagram, and *part A* is positioned lower in the Hasse diagram, the completion of assembly for part A will ***activate*** the assembly task for *part B*.

Therefore, we present our proposed solution, which revolves around a table that illustrates the activation relationships among PARTS, based on the transitivity of a finite partial order set. By utilizing the principles of topological sorting, we can execute assembly tasks from various initial points, aligning with the multiple assembly paths encountered in real-life situations. Drawing from the topological sorting algorithm, we introduce a template framework called CRUD for developing virtual assembly prototypes that accommodate multipath functionality. In this framework, *C* denotes core, encompassing the fundamental class models for system logic. *R* signifies resource, encompassing external assets such as CAD models, textures, and animations utilized for visual representation. *U* represents utility, classes, and function libraries for managing global operations and decoupling from core models. Lastly, *D* stands for data, which typically refers to a table that records the identification information of each component and the activation relationships between components. The structure of the CRUD framework is illustrated in Figure 2.

3.2 CRUD Framework

This section elucidates the components of the CRUD framework, with the *R* part referring to external resources and therefore not elaborated upon. The focus is on explaining the composition and functions of the *C*, *U*, and *D* sections.

3.2.1 Core

The core of this framework can be described using the ***PAM*** composition, which includes *Part*, *AttachPoint*, and *Master*.

Part is a general category that includes all individual parts. *Master* is the main part that is assembled using child parts (For example, in the case of the bench vise mentioned earlier, the base should be an instance of *Master*, while the other parts should be instances of *Part*). *AttachPoint* refers to a specific location on *Master* where all parts that can be assembled are located. There is also an abstract base class called *PartBase*, which internally stores information about specific components. ***PAM*** has an obvious triangular relationship, with *Part* and *AttachPoint* inheriting from the base class *PartBase*. The same pair of *Part* and *AttachPoint* shares the same set of part information. The *Master* needs to accommodate several instances of *AttachPoint*, forming a composite relationship between them. The *Master* derives from *Part*, and the assembly of *Master* instances also depends on *Part* instances.

3.2.2 Utility

The utility is responsible for supporting certain common operations that are not data-coupled. Here, an example of *QueryHelper* is given, which is responsible for globally querying *Part* instances and returning results. It also has the ability to activate parts. Utility is extendable.

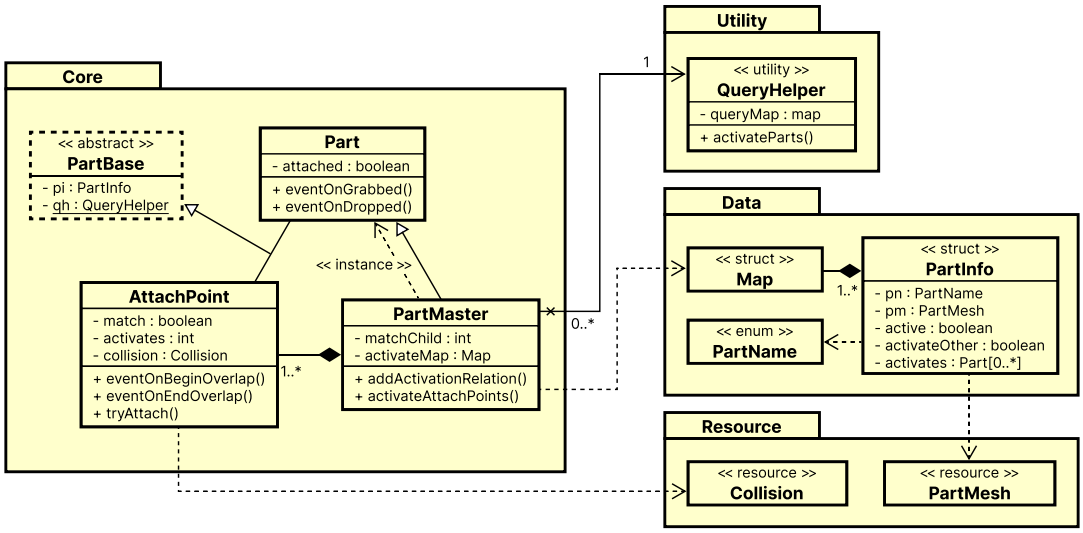


Figure 2: Class diagram representation of the CRUD framework.

3.2.3 Data

The primary component of the data section consists of a data structure that provides information about the parts and an activation table that outlines the assembly relationship before and after the part. Table 1 presents the relevant part information within the data structure, with each attribute accompanied by a corresponding description. The activation table is populated with data following this structure and serves as the central element of the entire framework, upon which the multipath assembly mechanism relies.

Table 1: Basic properties and their descriptions of the part infomation.

Properties	Type	Description
partName	enum	Friendly name of the part
partMesh	model	Model for visual representation
isActive	bool	If the parts is initially active or not
isActivateOther	bool	If other parts are activated by this
partActivates	Set	Names of parts activated by this

3.3 Mechanism

To illustrate the mechanism of this framework, let's consider a specific assembly node task within the assembly process. Figure 3 depicts the interaction sequence diagram between core's ***PAM*** composition and utility. When a user grabs an instance of *Part*, it triggers the `OnGrabbed` event for that instance. When the user overlaps the grabbed *Part* instance with an instance of *AttachPoint* on *Master*, it triggers the *OnBeginOverlap* event. If the *Part* instance matches and is in an active state, it binds the *OnDropped* event of the *Part* instance, preparing for subsequent delegation. When the user releases the *Part* instance in the overlapping state, it triggers the *OnDropped* event and delegates the *AttachPoint* instance to execute *TryAttach*.

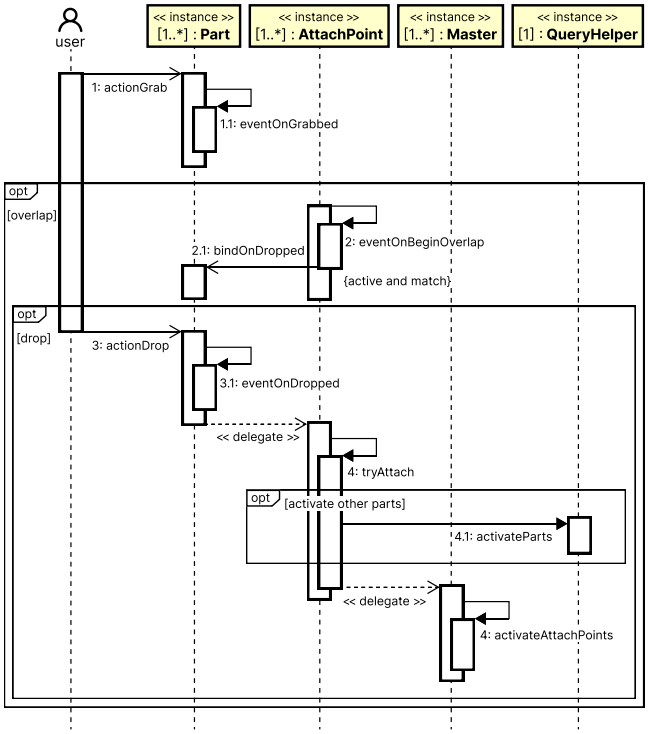


Figure 3: Sequence diagram represention of the core machanism

Here is a detailed explanation of *TryAttach* and the subsequent process. The pseudocode for the *TryAttach* process is shown in Algorithm 2 . First, it attaches the *Part* instance to *Master* as a child element. Then, it adds physical constraints to limit degrees of freedom for both elements. Finally, it removes the grab component from the *Part* instance. If this step can activate other *Part* instances, it calls the *ActivateParts* function using the *QueryHelper* reference.

Algorithm 2 TryAttach

Require:
P Current being assembled Part,
M Current assembling Master,
Q Reference to the QueryHelper.
Result: P is fixed onto M as a child with physical constraints. The relevant parts and attach points are also activated.

```
1:  procedure TRYATTACH(P,M,Q)
2:      if match then
3:          Attach P → M as child;
4:          Set Physics constraints between P and M;
5:          Remove Grab component of P;
6:          if P.activateOtherParts = true then
7:              Q.ACTIVATEPARTS(P.name);
8:          end if
9:          M.ACTIVATEATTACHPOINTS(P.name);
10:     end if
11: end procedure
```

The pseudocode for the *ActivateParts* process is shown in Algorithm 3. It uses the activation component name of the *Part* instance as the key to query in map and sets the *active* state to true for all returned instances. Then, the *AttachPoint* instance delegates the execution of *ActivateAttachPoints* to the *Master* instance, which activates the next assembly location points. The key represents the name of the *AttachPoint* instance, which is used to find matching objects in map. The process then sets the *active* state to true for all of the results. The pseudocode for the *ActivateAttachPoints* process is shown in Algorithm 4.

Finally, the *AttachPoint* instance is destroyed. With this, the assembly node task is completed, and the next assembly node task can proceed. This is the core mechanism that enables the framework to support multipath assembly.

Building the necessary attachment operations for assembly requires a clear hierarchy of the assembly. In more complex scenarios, people might consider assembling separate subassemblies. This framework also supports a two-level assembly structure, where only the entire subassembly composed of `Master` and its various child elements needs to be transformed into a new `Part` instance.

Algorithm 3 ActivateParts

Require:
P Newly assembled part.
Result: Following the completion of P, the subsequent step's part is activated.

```
1:  procedure ACTIVATEPARTS(P)
2:      for all i ∈ P.activates do
3:          for all r ∈ R, R ← queryMap[i] do
4:              r.active ← true;
5:          end for
6:      end for
7:  end procedure
```

4 IMPLEMENTATION

This framework can be implemented as a plugin or toolkit and can be applied in 3D engines such as Unity 3D or Unreal Engine. The end-users (i.e., developers) of this framework can import the plugin

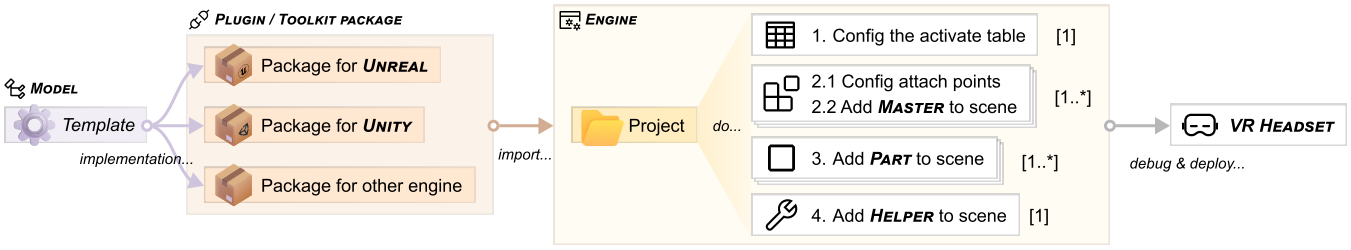


Figure 4: A reference pipeline for prototyping a system using the CRUD framework.

Algorithm 4 ActivateAttachPoints

Require:
A Current attach point being overlapped.
Result: Following the completion of A, the attach points for the subsequent step are activated.

procedure ACTIVATEATTACHPOINTS(A)
  if $A.activates = 1$ **then**
    $r \leftarrow activateMap[A]$;
    $r.active \leftarrow true$;
  else if $A.activates > 1$ **then**
    $n \leftarrow A.name$;
    for all $k.name = n, k \in K, K \leftarrow activateMap.keys$ **do**
      for all $r \in R, R \leftarrow activateMap[k]$ **do**
        $r.active \leftarrow true$;
      end for
    end for
  end if
end procedure

into their specific projects and utilize it to build a multipath virtual assembly. The following are the four steps required for developers to prepare and utilize this framework:

(1) Configure the activation table.
(2) Configure *AttachPoints* for each *Master* instance, and then add them to the scene.
(3) Add all *Part* instances to the scene.
(4) Add the single instance of utility classes to the scene.

With these steps, all the preparation work is completed.

This framework is extensible. For advanced users, custom function libraries, core class models, and activation table structures can be defined and developed based on project requirements. Figure 4 illustrates the above process

Unreal Engine 4 is the fourth generation of the Unreal game engine developed by Epic Games. The software includes a distinctive visual scripting system called *Blueprint*, which greatly lowers the barrier to entry for programming. This allows entry-level developers to implement basic functionalities in a short period of time. In recent years, UE4 has been increasingly applied in practical projects by animation and architectural companies. Considering the user-friendly interface, this paper uses Unreal Engine as an example to demonstrate the implementation of this framework.

5 End-user Development

The swerve drive module, which integrates drive motors, steering motors, reducers, and other components into a compact mechanical structure, was selected as the focus for the virtual assembly training system. The complex design of the swerve drive module presents a significant challenge for first-year undergraduate students when it comes to understanding its assembly process. In this paper, a virtual assembly training system is developed using the proposed CRUD framework, and the swerve drive module of FIRST Robotics Competition is used as the training material [2]. The virtual assembly system is deployed on the Oculus Quest 2.The objective is to validate the effectiveness and generality of the framework in complex scenarios, while assisting university students in comprehending the fundamental principles of complex mechanical structures and acquiring proficiency in the basic assembly process of the swerve drive module. Figure 5 shows a comprehensive view of the swerve drive module assembly.

5.1 Preparation

Commencing with the project setup, begin by creating an empty project using the VR project template provided by Unreal Engine. The version used throughout this paper is Unreal Engine 5.1 [1].

Subsequently, incorporate the Unreal implementation of the CRUD framework outlined in this paper as an Unreal content plugin within the established project. Following the import process, the plugin's content should comprehensively cover the key elements of the CRUD framework, as depicted in the Figure 6 below. Descriptions for each file can be found in Table 2.

5.2 Configuration

To illustrate the specific configuration, we will use the example of the wheel sub-assembly within the swerve drive module, as shown in Figure 7.

5.2.1 Configure the activation table

Before configuring the activation table, developers need to enumerate the names of all parts involved in the assembly. Figure 8 outlines

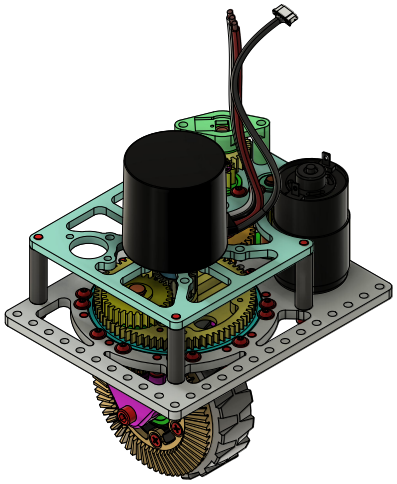


Figure 5: Assembly structure of the swerve drive module.

Table 2: Contents of the CRUD plugin, along with file types and descriptions.

Module	Asset	Type	Description
Core	BP_AttachPoint	Blueprint Class	Implementation of AttachPoint
	BP_Part	Blueprint Class	Implementation of Part
	BP_PartBase	Blueprint Class	Implementation of PartBase
	BP_PartMaster	Blueprint Class	Implementation of PartMaster
Data	F_ActivateMapStruct	Structure	Strucure expressing the mapping of activation relationships
	F_PartDataStruct	Structure	Structure storing infomation about parts
	E_PartName	Enumeration	Enumeration of all part names in the assembly tasks
	DT_PartDataTable	Data Table	Table containing infomation about all parts in the assembly task (Activation table)
Utils	BP_QueryHelper	Blueprint Class	Implementation of QueryHelper

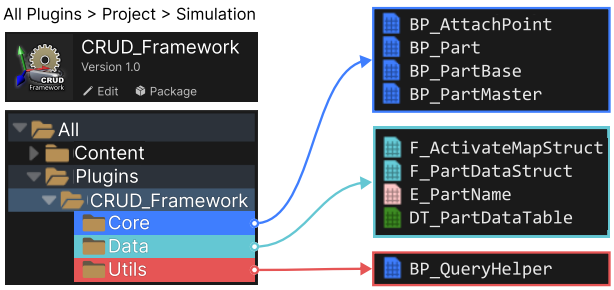


Figure 6: Contents of the CRUD framework plugin.

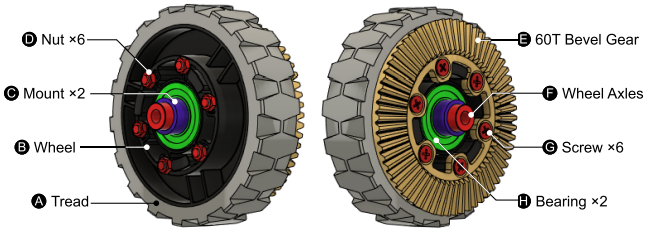


Figure 7: Wheel Sub-Assembly of the swerve drive module.

the process of configuring the activation table. For each part, create a new record using its enumeration name as the key. Then, set its activation status and select its visual model from the asset library. To indicate activation relationships with other parts, simply drag and drop the corresponding part record into the activation field. Additional fields can be added to accommodate more subordinate activating parts.

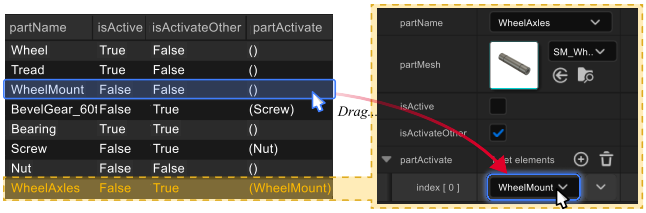


Figure 8: Operations for configuring the activation table.

5.2.2 Configure the master

For the wheel sub-assembly, it is intuitive to designate the hub as the `Master`. Therefore, within the outliner of the `BP_PartMaster` instance, add the `AttachPoint` components, as illustrated in Figure 9. The `AttachPoint` corresponding to the tread position is highlighted in blue in the viewport for visual reference. Concerning the activation relationships of `AttachPoint`, include map relationships in the config panel. Then, drag and drop the `AttachPoint` components from the outliner into their respective fields, serving as both the *key* and *value* in the activation map. Given that the structure of the value field is a set, it allows for one-to-one and one-to-many activation mappings.

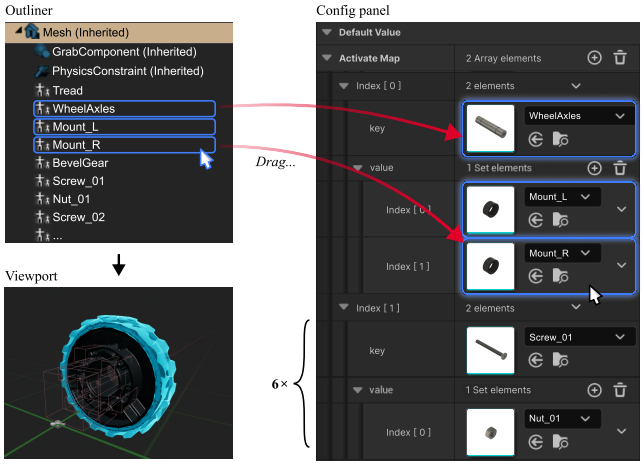


Figure 9: Configuring the AttachPoint components for the Master.

5.3 Add to Scene and Deploy

Upon completing the configuration of the `Part` and `Master` instances for the assembly training, add them to the final interactive scene. Additionally, include an instance of `QueryHelper` in the scene. Once these steps are completed, deploy the project and make it available for use on VR headsets. Our virtual assembly training system for Swerve drive module has been successfully deployed and tested on the Oculus Quest 2.

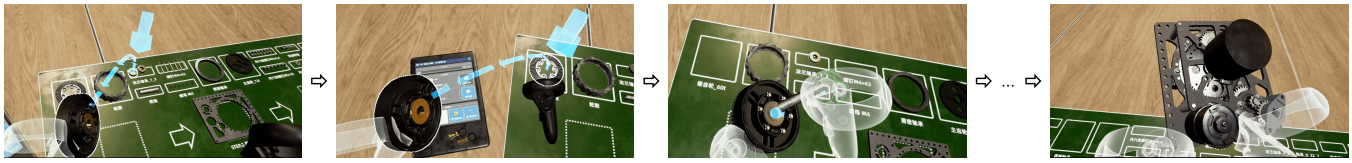
6 Experimental Validation

6.1 Participants

The experiments were conducted at the School of Design, South China University of Technology. Sixteen students with no prior experience in using VR and no prior knowledge of the swerve drive module were recruited as participants. There were 8 female (50%) and 8 male (50%) participants, aged between 19 and 24 years. Before the start of the experiment, participants were given instructions regarding the tasks they needed to complete and the maximum time for the entire experiment (a total of 120 minutes). Participants also received a detailed introduction on how to use the Oculus Quest 2.

6.2 Tasks

The experiment consisted of two tasks: assembling the swerve drive module using the VR system and assembling a physical model



(1) Assembly of flange bearings. (2) Assembly of the bevel gear. (3) Assembly of the wheel shaft. (n) Complete.

Figure 10: Using Unreal Engine 4 to implement the CRUD framework and create virtual assembly training content. The picture shows the process and final result of assembling swerve drive module.

of the swerve drive module. Before starting, participants were randomly divided into two groups, labeled as Group A and Group B. Group A, comprised of 5 females and 3 males, first completed the virtual assembly task using the VR system and then assembled the physical model. Group B, which comprised of 3 females and 5 males, first assembled the physical model and then completed the virtual assembly task using the VR system. Participants were required to complete the assembly tasks as quickly as possible. After completing the first assembly task, participants had a suitable rest period before proceeding to the second assembly task. Figure. 10 depicts the step-by-step process of assembling a swerve drive module in VR. After conducting the assembly training experiment, we recorded the time taken by all participants to complete the tasks.

6.3 Results and Discussion

The disparity in the duration required to complete the tasks is depicted in Figure. 11. The time taken to complete tasks using the VR system was significantly lower than assembling the physical model. This could be attributed to the relatively easy learning curve of the VR system, which offers an advantage by simplifying less critical steps (such as screwing screws), while the actual physical assembly necessitates a methodical and meticulous approach. The statistical chart not only demonstrates the efficiency of virtual assembly in terms of the required time but also shows that the results are less dispersed compared to assembling the physical model. Some participants spent a significant amount of time assembling the physical model. On the other hand, the effectiveness of virtual assembly is also evident from the experimental arrangement of different groups. Group A performed physical assembly first and then virtual assembly, resulting in more time being spent on physical assembly compared to Group B. However, Group B had prior experience with virtual assembly, resulting in generally less additional time spent on physical assembly.

In addition, we primarily validate the versatility of the framework in complex scenarios by observing the diversity of assembly sequences among participants. Given that the omnidirectional wheel involves up to 105 component instances, we use the WHEEL subassembly as a representative example. The most commonly adopted assembly sequence is determined through experimental observations:

$$B \to A \to H_{\times 2} \to E \to G_{\times 6} \to D_{\times 6} \to F \to C_{\times 2}$$

For the assembly of screws and nuts, the recommended sequence involves first assembling all the screws before uniformly attaching the nuts. Another feasible approach is to assemble one set of screw and nut combinations before moving on to the next set, as adopted by many participants, with the assembly sequence:

$$B \to A \to H_{\times 2} \to E \to (G \to D)_{\times 6} \to F \to C_{\times 2}$$

Some participants choose to assemble the tread in a later sequence rather than at the beginning, resulting in the following sequence:

$$B \to H_{\times 2} \to E \to (G \to D)_{\times 6} \to A \to F \to C_{\times 2}$$

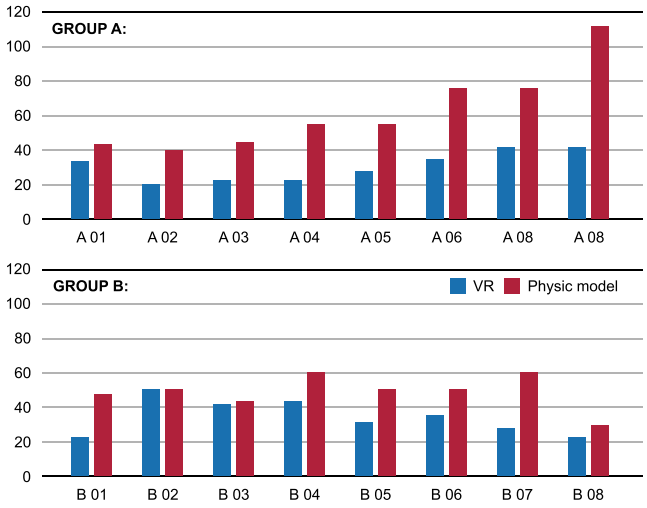


Figure 11: The temporal distribution of the completion of the task by the two groups of participants.

According to experimental results, seemingly "random" assembly sequences are also achievable as long as the activation relationships are satisfied. For instance:

$$B \to E \to (G \to D)_{\times 2} \to H_1 \to G_{\times 4} \to D_{\times 4} \to H_2 \to A \to F \to C_{\times 2}$$

The framework's versatility in accommodating multiple sequences in complex scenarios has been successfully validated, as assessed by the experimenters.

7 Limitations and Future Work

Although the effectiveness and generality of the CRUD framework have been experimentally validated, it still has certain limitations.The CRUD framework is particularly suitable for building prototypes that support multi-path assembly training for household products or small studio-level mechanical products, such as furniture, appliances, and robots. Generally, it works well at this level of complexity because these products have a clear assembly structure. However, for large-volume assemblies such as automobiles, using CRUD to build assembly training prototypes may not be as intuitive, as they typically have three or more levels of assembly. While configuring the assembly activation relationship table is typically done through a graphical interface (often involving drag-and-drop operations, especially in engines like Unreal Engine), it may not be considered easy.

Our framework is currently focused on training for non-large assembly purposes, making it very suitable for generating clear multi-path assembly sequences to realistically reproduce real-world assembly processes. Multi-path assembly of products such as bench vises, gearboxes, or swerve drive modules can be easily accomplished using the CRUD framework. In the implementation of the

Unreal Engine 4, developing assembly prototypes is straightforward. During the testing experiment of the framework, three undergraduate students majoring in fields other than computer science discovered that they could effectively use the framework to develop virtual training prototypes that support multi-path assembly. They were able to do this after a short learning period, and two of them even expressed interest in pursuing further development. However, specific implementation on other engine platforms, such as Unity3D, has not been done for this framework.

We proposed a framework, CRUD, for rapidly building virtual assembly prototypes that support multi-path assembly simulations of real-world assembly scenarios. Compared to existing methods, our framework offers unique advantages. Firstly, our framework allows end-users (developers) to create virtual assembly sequences based on simple configurations. This means that sequences can be customized for various projects and requirements, significantly enhancing flexibility. Secondly, multi-path assembly adheres to real-world physical principles, ensuring that learners' experiences in the virtual environment align with reality. This enhances the practicality and effectiveness of training. Most importantly, our framework improves system reusability, reduces development costs and time, making it applicable to a wider range of learning scenarios. We believe that supporting multi-path assembly into virtual training is a small step towards creating more realistic and practical virtual simulations.

One direction for future work is to focus on the adaptability of large assemblies. While this work contributes to the development of multi-path assembly prototypes for typical assembly models, the framework's upper limit of applicability, or its ability to handle large-scale use cases, remains an unavoidable question. Another direction is to focus on the specific implementation of the framework, including implementing it on other engines such as UNITY3D as a planned task. Additionally, we will address user experience concerns, including improving the convenience of configuring the assembly activation relationship table and enhancing the overall workflow to increase efficiency.

REFERENCES

[1] Release unreal engine 5.1.1 · epicgames/unrealengine.

[2] First robotics competition. *Wikipedia*, Oct. 2023.

[3] H. A. Adas, S. Shetty, and S. K. Hargrove. Virtual and augmented reality based assembly design system for personalized learning. pp. 696–702, 2013.

[4] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics*, 22(3):828–837, July 2003.

[5] A. M. Al-Ahmari, M. H. Abidi, A. Ahmad, and S. Darmoul. Development of a virtual manufacturing assembly simulation system. *Advances in Mechanical Engineering*, 8(3):168781401663982, Mar. 2016.

[6] M. V. A. R. Bahubalendruni and B. Putta. Assembly sequence validation with feasibility testing for augmented reality assisted assembly visualization. *Processes*, 11(7):2094, July 2023.

[7] A. K. B. G. Bharathi and C. S. Tucker. Investigating the impact of interactive immersive virtual reality environments in enhancing task performance in online engineering design activities. Jan. 2016.

[8] F. Dyck, M. Pilates, L. Masjutin, and J. Stöcklein. Physically realistic simulation of mechanical assembly operations in a virtual reality training environment. Lecture Notes in Networks and Systems, pp. 177–188, 2022.

[9] M. Eswaran and M. V. A. R. Bahubalendruni. Challenges and opportunities on ar/vr technologies for manufacturing systems in the context of industry 4.0: A state of the art review. *Journal of Manufacturing Systems*, 65:260–278, Oct. 2022.

[10] J. Gaarsdal, S. Wolff, and C. B. Madsen. Real-time exploded view animation authoring in vr based on simplified assembly sequence planning. pp. 667–668, Mar. 2023.

[11] D. Kalkofen, M. Tatzgern, and D. Schmalstieg. Explosion diagrams in augmented reality. pp. 71–78, Mar. 2009.

[12] K. H. Rosen. *Discrete Mathematics and Its Applications*. 7th ed ed., 2012.

[13] P. A. Winkes and J. C. Aurich. Method for an enhanced assembly planning process with systematic virtual reality inclusion. *Procedia CIRP*, 37:152–157, 2015.