VIRTUAL ENVIRONMENT FOR MECHANICAL ASSEMBLY SIMULATION AND ITS APPLICATION

by

Yizhe Chang

A DISSERTATION

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Yizhe Chang, Candidate

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2016

A VIRTUAL ENVIRONMENT FOR MECHANICAL ASSEMBLY SIMULATION AND ITS APPLICATION

ABSTRACT

Many methods of simulation of mechanical assembly processes have been researched and developed. They provide a good and vivid estimation of an assembly, thus facilitating the design of mechanical products. However, most of them only take into consideration the geometric shape, kinetic properties and stress distribution of the assembled product but neglect the assembly procedure and human interactions during the assembly.

In this project, an immersive virtual environment (VE) for assembly simulation was developed. Based on 3D game technology, this VE not only provides an immersive simulation environment, but also allows multiple users to work simultaneously on the same assembly to simulate collaborative assembly processes. In order to facilitate the assembly simulation, an assembly representation, which is adapted from computer-aided modeling and design research, is described in this document. This adaptation uses major computer-aided modeling and design concepts, such as features, feature association, hierarchical assembly, etc., to enable VE authors to conveniently make their own assembly simulations for various purposes.

Garry's Mod (GMod), a game engine sandbox, was selected in this project as the platform for implementing a VE for assembly simulation. Educational gear train VE laboratories were authored based on the assembly representation described in this document by using GMod. Usability evaluations of the laboratories on students majoring in mechanical engineering show that students are able to complete such laboratories regardless of their gaming background or lack thereof. Learning effectiveness evaluations indicate that the laboratory can marginally improve students' understanding of relevant topics.

In addition, the Microsoft Kinect sensor was explored for enhancing the feel of immersion for users of this VE. By gesture and speech, users can generate assemblies in the VE without a keyboard and mouse. The operations of gesture and speech-based assembly in the VE are defined in this document.

Author: Yizhe Chang

Advisor: Sven K. Esche

Date: June 30, 2016

Department: Mechanical Engineering

Degree: Doctor of Philosophy

Acknowledgement

In my journey of studying in Stevens Institute of Technology, I received a lot of help from people around me. My advisor, Dr Sven K. Esche, devoted hours and hours in guiding me, an immature student, in all academic aspects: course studying, writing, oral presentation, research topic forming, research methodology, teaching, etc. I gained my confidence of being qualified for the Ph.D. title not only because the committee’s approval signatures; but also, more importantly, the skills he trained me and the philosophy he inspired me, which cannot be reflected by a dissertation.

I sincerely thank my dissertation committee members, Dr Constantin Chassapis, Dr James E. Corter, Dr Frank T. Fisher, and Dr Sophia Hassiotis, who provide many advices for my dissertation. As my study has broad topics, their professionalism is the key for letting me finish this dissertation.

My collaborators, Dr El-Sayed S. Aziz, Dr Mingshao Zhang, Mr Zhou Zhang, Mr Shi Bai, and Mr Junjun Ding, played vital roles in supporting my studying. Besides our pleasant collaborations, their optimism and tolerance, are big momentums for me.

In addition, I would sincerely thank all staffs in Stevens Institute of Technology for their helps.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Background

Although the definition of the term virtual environment (VE) has not reached consensus, VEs are generally regarded as computer simulations that reconstruct real-world scenarios, with which human beings can naturally interact in various ways (such as via video, audio, haptics, etc.) [1] [1].

Unlike major professional computer simulations, such as finite element analysis, machine dynamics analysis, circuit analysis, etc., which focus on the accuracy of the simulated results, VEs are designed to present complete processes of real-world activities that allow individuals or groups to participate. Therefore, VEs are not only able to simulate physical phenomena that are included in the activities in a real-time manner, but they also allow real-time human-machine interaction. This interaction is bidirectional. On one hand, simulation results (mainly visual and audio effects) are delivered to humans in a timely manner through output devices such as monitors and loudspeakers. On the other hand, humans are able to affect the simulation at any time by giving commands through input devices such as keyboard and mouse.

VEs are widely applied nowadays. From schools, to museums, factories, the military and medical clinics, VE applications are becoming increasingly ubiquitous. These VE applications simulate various scenarios to serve purposes such as education, training, remote control, collaborative design, etc., by providing users a feeling of immersion into these scenarios without physically exposing them. They are therefore safe and low-cost. VEs also exhibit advantages in completing remote collaborative activities via computer networks. For instance, virtual classrooms are utilized for online lectures so that geographically separated students and lecturers can ‘sit under the same roof’ to engage in

[1] As in most cases where the terms virtual environment (VE) and virtual reality (VR) are used interchangeably, in this document, the term VE is used to describe the environments that are implemented by the author of this document; the term VR is used in the literature review where VR is applied in the original text.

pedagogical activities. Such VEs can be even more valuable when the simulated scenarios contain dangerous activities that are not suitable for inexperienced individuals, such as fire-fighter or pilot training. VEs can also simulate rarely-happening scenarios that cannot be simulated in the real world, such as military action.

1.2 Applying VEs for mechanical assembly processes

VEs can also be utilized to simulate mechanical assembly activities. They can provide immersive graphics effects by presenting the assembly environment (e.g., an assembly plant), models of mechanical parts, and tools. Besides the graphics, VEs usually facilitate physics simulations that model effects such as gravity, collisions, and sometimes even damage. More importantly, VEs are interactive and may allow people to complete assembly tasks by controlling human-like avatars.

VEs for mechanical assembly applications can be found in the fields of industrial design, assembly training and engineering education. In industry, VEs are powerful tools for decision making processes by providing new assessments during assembly planning and assembly design review. Since the costs associated with assembly processes often constitute a large portion of the product manufacturing cost, assembly planning is critical [2]. A virtual assembly environment can function as an integrated expert system that, on one hand, maintains the advantages of various computer-aided assembly planning (CAAP) systems, and on the other hand, addresses the human-machine interaction [3]. For design review, VEs can be used to examine the validity of the designed assembly process. The flow of material, the sequence of assembly, the paths of component movement and the coordination of personnel can be planned and verified in a VE, which may uncover errors in the design stage, reduce the cost of prototyping, and avoid exposing people to uncertain environments in the early implementation stage. Furthermore, VEs represent a powerful tool for training novice personnel if the real assembly environment contains dangerous elements. In engineering education, VEs can be useful since, on one hand, they can provide scenarios that academic institutions cannot afford to provide in the real world; and on the other hand, they can simulate ‘idealized’ environments in accordance with pedagogical goals that help students to understand some theories without being distracted by irrelevant factors such as missing parts or malfunctioning tools.

1.3 Requirements of VEs for mechanical assembly activities

In order to simulate mechanical assembly activities through VEs, many factors should be considered. They can be classified into three major fields:

1. Mechanical modeling
2. Assembly process simulation
3. Immersive experience

For mechanical modeling, there are issues such as the geometries of mechanical parts, the topology of assembly structures, and the kinetics analysis of the assembly, etc. For assembly process simulations, issues such as human-computer interaction for the simulation of assembly operations, human-human interaction, etc. arise. In order to provide an immersive experience to VE users, ambient environments for the specific assembly scenario (e.g., assembly laboratories or assembly plants) are required.

The topics relating to the geometrical and topological modeling of mechanical assemblies have been addressed by CAD research for a long time, and CAD software for product design is being widely applied nowadays. CAD software covers the functionalities of parametric geometry modeling, constraint-based assembly modeling, creating engineering drawings, kinematic analysis, and assembly sequence planning. Such CAD software has become the foundation of concurrent engineering, which lowers the costs, improves the product quality, and reduces the product delivery time.

The simulation of assembly activities has also received attention in the past. In order to enhance the experience of assembly operations in VEs, besides using keyboard and mouse, researchers have tried out various input devices. A widely researched example is haptic devices. In order to reinforce communication and cooperation during an assembly process, collaborative systems are introduced, although most of them target assembly design and focus on file sharing.

Sometimes referred as virtual reality (VR), VE technologies have also been deployed for scenario simulation in previous research. From desktop virtual reality to immersive virtual reality, ambient environments provided for mechanical activities can meet different levels of needs.

However, current research usually focuses only on one or several factors mentioned above. There is no effective integration of all these factors to provide a fully functional environment for assembly simulation. On one hand, for most computer-aided designs, simulating assembly activities and providing an immersive environment are not necessary, since for these applications, the functionalities and specifications of products are addressed. In some applications, a VE can contribute nothing but distract the designers.

For mechanical assembly activity simulations, researchers generally concentrate on the user experience when manipulating input devices while not systematically integrating the mechanical assembly modeling. In some papers, ambient environments are integrated into the virtual reality experience. However, most of these integrations are meant for visual purposes only. That is, there is no interference between the avatar(s) and the ambient environment. Avatars can usually go through walls in these applications. The usage of immersive VEs for various purposes, such as virtual tours, virtual training, virtual manufacturing, etc. has also been researched. However, attempts that were made for mechanical assemblies are limited so far. This is attributable to the fact that mechanical assemblies involve various types of individual parts that are combined into complex systems, which most VEs do not take into consideration.

This gives rise to the question whether there is any method for effectively integrating all issues related to mechanical modeling, assembly process simulation and creating an immersive experience. In this proposal, a VE for mechanical assembly processes is presented that was implemented based on the platform of a game engine. Utilizing the rendering, physics, audio, human-machine interaction, and artificial intelligence (AI) modules of this game engine and combining them with geometrical and topological modeling of mechanical assemblies, this VE is able to support the simulation of assembly processes.

However, in order to author VEs for mechanical assembly activities, besides developing general VE elements such as a room, furniture, chairs, etc., elements such as geometrical modeling of mechanical parts, topological modeling of assembly structures, regulating of and reasoning about assembly sequences, and simulating mechanical dynamics have to be considered. Also, the shortcomings of VE technologies such as inaccurate simulation functionalities and non-user-friendly human-machine interfaces impose multiple problems in VE authoring. A successfully implemented VE must overcome these challenges.

In the next section, requirements of VEs for simulating mechanical assembly activities are discussed. In Section 1.4, game engines are proposed to be used as platforms for such VE authoring.

1.4 Game engines as development platforms for mechanical assembly VEs

State-of-the-art video games attract players by many elements, such as fascinating backgrounds and exciting adventures, high-fidelity audio effects, near-real 3D graphics, reasonably realistic real-world physics simulations as well as the strong engagement of the players. Although from a content perspective, these games can be classified into multiple categories such as role-playing and real-time strategy, from a technical perspective, they share many common features. The infrastructure of video games has three layers: the system layer, the game engine layer and the game play layer. The system layer includes the hardware and its drivers, the operating system and some fundamental programming interfaces. The system layer provides the foundation of video games. The game engine layer includes the physics, rendering, audio and basic AI engines. In online multiplayer games, communication modules are also included to enable interactivity. The game engine layer supports game developers by providing ready-to-use functions for complex gaming effects. Finally, the game play layer includes all the characteristics that are offered to the players, such as avatars, challenges, tools, weapons, etc. The development of the game play layer requires combined efforts of computer technicians and, more importantly, artists.

Taking advantage of many VE authoring tools that are provided by game engines (e.g. modeling tools), designers of mechanical assembly simulations can be freed from repetitive tasks such as assembly model building, laboratory environment authoring, etc. Instead, the designers can focus their efforts on composing assembly scenarios, planning experiments and programming non-player avatars (NPC). Although most game engines are designed for entertainment purposes, the basic functions for graphics, physics simulations and story plots are capable of supporting the design of educational computer games. In addition, nowadays, many game engines are not game specific but are developed to support a wide range of games. Based on their 3D graphics and real-world physics simulations, such game engines not only allow for the development of game environments that give the users a feel of reality and being immersed, but they are also designed for ease of development based upon them. A good example is the ‘Source’ game engine, which fueled the development of famous games such as Half-life and Counterstrike, and includes the ‘Source’ Software Development Kit (‘Source’ SDK) which enables game developers to build their own games.

1.5 Using a game engine for authoring a VE

Adapted from computer-aided design research, a mechanical assembly representation that is specifically adjusted for simulating assembly processes in a VE, which uses a popular game engine as the development platform, is described in this document. This VE gives assembly designers a unified model for describing mechanical assemblies for assembly simulations and allows staff to collaboratively generate mechanical assemblies. A gear train assembly education scenario, which was authored based on the VE and the assembly representation, was evaluated for its usability and education effectiveness on more than 200 students over a period of 3 years. In addition, in order to enhance the feel of immersion during the generation of assemblies, a gesture- and voice-based human computer interface was also piloted by integrating a Microsoft Kinect sensor into the game engine based VE.

The structure of this document is as follows: Chapter Chapter 2 is a literature review regarding the topics of assembly representation, VE/VR and their applications, the Microsoft Kinect for non-entertainment use, and state-of-the-art educational remote laboratories. Chapter Chapter 3 illustrates an adaptation of an assembly representation for VEs based on game engines. Also, the procedure of the assembly simulation is discussed. Chapter Chapter 4 explains the requirements and the advantages of using a game engine for authoring a VE for mechanical assembly simulations. Chapter Chapter 5 demonstrates an application of this VE for educational purposes. Two VE-based laboratories are introduced and evaluations of usability and learning effectiveness were conducted. In Chapter Chapter 6, a Kinect interface for this VE is explained, followed by conclusions.

Chapter 2 Literature review

2.1 Assembly representation

2.1.1 Methods for geometry description of mechanical parts

Mechanical parts are the fundamental elements of assemblies. A part is a functional element that cannot be further mechanically divided (i.e. disassembled) as such further division would cause the part to lose its functionality. A part can be as simple as a flat plate or as complex as an engine piston. Compared to an assembly or a sub-assembly, a part must be replaced in its entirety when it breaks, while in contrast some portions of an assembly/sub-assembly could potentially be reused. Therefore, to describe an assembly, the first step is to find a method for describing a part.

Certain mechanical parts can be approximated as being rigid since their deformation is negligible even under external loads. Other mechanical part must be treated as deformable since their shape changes appreciably even when moderate external loads are applied. A part can be modeled by various methods using computers. Among these methods, decomposition modeling, constructive modeling, and boundary modeling are mostly utilized [4].

Decomposition modeling is a straightforward method that represents a part by the 3D space that the material occupies. The part is decomposed into small 3D pieces with certain shapes. The information of each piece, such as type of the shape, dimensions, spatial coordinates, etc. are recorded. Based on the shape of the small pieces, this method comprises several schemes, such as exhaustive enumeration, cellular decomposition and space subdivision. Among them, space subdivision is most efficient as it requires the least amount of storage space for a given part. A successful example of space subdivision is the octree representation [5] [6].

Constructive modeling uses Boolean operators to form parts from primitive shapes. The Constructive Solid Geometry (CSG) approach is one of the most commonly applied modeling methods in this category. A part represented by CSG can be formed by operations such as union, intersection and subtraction of primitives such as box, cylinder, cone, sphere, etc. [7].

Nowadays, boundary modeling is the most commonly used method to describe a part. In the boundary representation method, a part is described by its bounding surfaces, which usually are plane polyhedrons. These polyhedrons are further described by their vertices.

In modern computer graphics, the most common geometries that graphics cards process for rendering are polygonal primitives [8]. That is, no matter how a part is modeled and stored, it must be converted to triangle-shaped boundary models for rendering purposes. Therefore, it is straightforward to model a part directly by this boundary modeling method wherein the part is represented by triangular meshed surfaces. The position coordinate, along with other properties (e.g. normal vector, texture coordinate, light effects) of each vertex, are recorded.

In all 3D computer game environments, solid models are described by triangular meshes. Therefore, to render a mechanical part in a game environment, the part must be converted into a triangular boundary representation, no matter by which method it was originally described. The conversion is supported by most current 3D design software such as 3ds Max, AutoCAD, SolidWorks, Pro/Engineer, etc. In the former two software packages, users can actually directly manipulate the boundary models during the design process while in the latter two the boundary models are not directly accessible to the user.

2.1.2 Feature-based modeling

Parametric and feature-based design is not new. Unlike traditional geometry modeling methods, which require users to create a 3D geometry in a tedious vertex by vertex fashion, the parametric feature-based method facilitates high-level design by providing specifications to pre-defined primitives (e.g. cylindrical hole) and pre-defined actions (e.g., extrude, cut).

In the engineering design field, feature-based modeling exhibits advantages mainly in two aspects [9]:

1. Complete definition of product: Not only the geometry of a product can be defined, but also other design elements such as tolerances, surface finishes, material specifications, surface treatments, etc. can be included in the model.
2. Definition of geometry at a high level: The manipulation of vertices and operations on primitives are replaced by highly abstracted geometrical (form) features. The time of creating designs is reduced.

2.1.3 Topological representation of assemblies

Although assemblies vary to a very large extent in their appearances and functions, from a topological point of view, they are made from the same structures. They are composed of parts, which include features (e.g. surface, curve, threated hole, etc.). Parts are linked by joints to form sub-assemblies. Sub-assemblies are then linked by joints to form assemblies.

The representation of such topologies was studied and implemented in many forms. There exists research on how to model an assembly using relational databases. For instance, a database that contains the tables of parts, assemblies, assembly tools and assembly machines has been created [10].

A mereotopological representation was used to describe such a concept [11]. Mereotopology uses logical symbols and sentences to express the relationships of regions and entities such as features, parts and assemblies that occupy a certain region [12] [13].

However, for assembly design, mereotopological notions, definitions and primitives, which are represented by logical symbols, “lack the universality of the semantic definitions” [14]. Therefore, in order to apply mereotopological principles for developing CAD software, the Semantic Web Rule Language (SWRL) is widely utilized by the research community to describe mereotopology terms [15].

Another common topological representation of assemblies is by linked graphs. The shape of a part can be modeled by geometrical features, and links between two parts can be modeled by constraint features. This improves the work efficiency considerably compared with traditional low-level geometry design. At different design phases, these features are different. Multi-view feature modeling integrates all the features that are needed in different design phases of an assembly [16].

An advantage of this constraint-based representation is that it makes kinematics solutions convenient. Geometry solvers were developed for simulating the kinematics of assembly models by this representation. Research on solvers is ongoing. In order to define a geometry constraint problem, two fundamental sets must be included: a set of points *P* and a set of constraints *C* [17]. There are mainly two constraint solvers: equation solvers and constructive solvers [18].

2.1.4 Assembly representation

In an assembly, it is crucial to develop a model that represents the relationships between the parts. Such a model omits the detailed information of the assembly, such as the shape of the parts or the exact type of the joints, and instead concentrates on the logical connectivity of different parts. Building such a model is a proactive approach employed in Design for Assembly (DFA) and Design for Manufacturing (DFM) [19].

An assembly can be represented by lists of tuples. An assembly sequence table (AST), which contains tuples of possible assembly states, was created [20]. A virtual manufacturing lattice (VML) was used to store an assembly with a 4-tuple, <C, R, T, E> structure. In this 4-tuple, C is the composition element to store the geometry of a part, R is the relationship of a part to the other parts, T represents the trajectory of assembly, and E is the event control list to identify the states of the assembly [21].

The most obvious way to represent an assembly and its associated assembly sequence is by graphs. Connection graphs describe the associations of parts in a direct way, but they do not show the sequence of assembly [22] [23]. Directed graphs as well as ‘and/or’ graphs for representing assembly sequences efficiently have been described [24] [25]. These graphs first enumerate all possible assembly sequences and then remove those sequences that are not suitable due to mechanical or geometrical constraints.

However, an assembly is constrained not only by physical joints, such as welding joints, but also by spatial relationships. In order to represent this situation, a relationship matrix (RM) was created [26]. In that work, two types of part relationships were defined: physical joints and ’layout interference’, which represents spatial constraints.

2.2 Collaborative assembly design systems

Compared to the collaboration functions in other collaborative design systems, a major advantage of VEs is that they enable “face-to-face” collaboration. This section reviews the design research community’s exploration on collaborative design.

Many efforts on collaborative assembly design have been reported. In order to design the process of assembling a machine, different professionals, such as product manager, product architect, and designers with different expertise, are included. Problems related to the information sharing between the personnel may arise. The emergence of the Internet renders this problem even more severe since the designers may be geographically dispersed, the CAD software they use may be different, and the version control problem is amplified because making changes to designs has become much easier than before.

A Web-based collaborative assembly design system should have the following features [27]:

1. It should include a database that stores the information about components and sub-assemblies for the assembly. The information can be 3D models, material requirements, manufacturing procedures, etc.
2. It should offer communication among all teams and their members. The communication can be in multi-media format.
3. It should provide design tools (such as CAD software) and resources to authorized personnel. For instance, the project manager can authorize circuit designers to access NI Multisim (electrical circuit simulation software) through the collaborative system, but not authorize

them to use SolidWorks (mechanical design and simulation software), which is accessible to mechanical designers.

4. It should coordinate tasks and progress for concurrent engineering.

Among these many features, building the database of the components' and sub-assemblies' information for 3D model storage and communication has drawn the most attention in the past. In order to solve the issue that formats of 3D models created by multiple CAD software packages may be incompatible, the Standard for the Exchange of Product Model Data (STEP, ISO-10303) and the Extensible Markup Language (XML) were created and adopted by most Web-based collaborative assembly design systems.

A general method for using STEP as the converter to solve the issue of the mapping between product data management (PDM) systems and different CAD/CAM systems was devised [28] and a client-server model network system was proposed [29]. In this system, a STEP server is set up for storing all STEP-standard information pertaining to the mechanical components and sub-assemblies [30]**.** The client side is for designers, who can retrieve and update server-side information by using STEP supported CAD software. The server side can further add components such as coordination manager, geometry engine and constraint engine. These components can process STEP geometry data updated by clients, form assembly and sub-assembly STEP model files, and distribute them back to all related clients subject to coordination rules.

2.3 Using a VE for assembly simulations

2.3.1 Applications of VE technology

Also referred to as VR in the literature, the VE technology does not only feature immersive visual output (a display) but also addresses the interaction between users and computers [31] [32]. VEs are designed to be able to respond to user commands in a real-time manner. In order to enhance the immersive experience, novel I/O devices, such as stereo-vision glasses [33], human motion tracking sensors [34], haptic devices [35], etc. are used. Compared with 'professional' simulation applications such as stress analysis or motion analysis, VEs emphasize more the user experience they provide instead of precise simulation data.

Applications of VEs can be found in many fields. For entertainment purposes, virtual tours have been developed (e.g., virtual tours of museums [36]). When users attend a virtual tour of a museum, they can navigate through the museum, visit the collections or even ‘touch’ the collections. For training purposes, VEs have already been in practice for many years. Successful cases include pilot training [37] [38], surgical training [39] [40], etc. Traditionally, these types of training are either dangerous for trainees or expensive in practice. These applications therefore focus on the human-machine interaction with very specific purposes. For medical purposes, psychotherapy methodologies based on VR have been researched [41] [42]. These applications take full advantage of the immersive experience provided by VEs and provide a virtual world for patients.

VEs are also able to facilitate product realization [43]. Currently, related research covers the simulations of all stages of a product life cycle by VE technology. There are reports on the usage of VE technology for product realization, addressing topics such as virtual conceptual design, virtual manufacturing, engineering analysis, visualization of analysis results, collaborative VEs, etc. Compared to traditional CAD design software, VEs exhibit advantages by providing ambient environments, which may help designers to review the real-world application of the designed product in the context of the application or manufacturing scenario so that potential problems could be identified [44]. By using immersive VE devices, designers can further create products that better comply with ergonomics regulations. In the automotive industry, this design method has already been applied [45] [46]. For manufacturing, VE technology is said to be able to help in assessing the manufacturability of a design, estimating the cost of the product, the processing time of manufacturing, and even the product quality [47].

2.3.2 Simulating assembly processes using VE systems

There have already been many reports of successful simulations of manufacturing processes [48]. Commercialized software such as DELMIA can "create, optimize, and validate the assembly process in the context of its manufacturing setting" [49]. In addition, machining processes of a specific machine can also be simulated by VE systems, such as the trajectories of cutting tools [50] [51], the motions of different parts of a machine [52], the operation of a machine [53], etc.

The mechanical assembly process is a part of the product manufacturing process. Since the early 1990s, VE technology has been applied to create virtual assembly (VA) simulations in order to accelerate assembly planning [54]. Nowadays, virtual assembly simulations integrate multiple expert knowledge fields that traditional CAD software, which is designed for geometry and mechanical analysis, is incapable of handling. These expert knowledge fields include assembly time estimation, sequence planning, tooling, fixture, safety, ergonomics, etc. [3] [55].

There are several perspectives to classify existing VA systems. From a purpose perspective, applications that use computers to assist assembly-related engineering decisions through analysis, predictive models, visualization, and presentation of data, without physical realization of the product, can be categorized as VA applications [56]. From a user-experience point of view, virtual assembly systems should enable the users to assemble CAD models in an immersive environment, with a natural human-machine interaction interface [57].

Since the core purpose of VA is to facilitate the design, to evaluate a VA environment, besides its capability of processing geometries, some more subjective topics considered in the traditional assembly design method should also be assessed. In order to author a VA environment, the following features should be partially or fully implemented [58]:

1. Can the part be handled by one hand (or are two hands required to cope with the part's weight and dimensions)?

2. Are parts nested or tangled within one another[2]?
3. Are handling tools required?
4. Are there any obstructions of parts, tools or hands?
5. Is there any vision blockage?
6. Does maintaining the part's orientation or location during subsequent operations requires holding the part down?
7. Is it easy to position or align parts?

In order to addresses these application-related questions, various VA systems have been explored by different research groups. Different levels of VEs were included in their systems. Some of them use desktop VR, which requires merely a basic computer with conventional I/O devices such as keyboard, mouse and monitor, while some others integrate non-conventional I/O devices such as body motion sensors, gesture recognition gloves, stereotyped glasses, etc. to enhance the immersion of the user in the VR. For the former type of systems, the hardware is quite affordable and the product can be easily distributed. For the latter type, assembly procedures can be simulated in an enhanced immersive fashion for professionals. In this proposal, the desktop VR is mainly used.

2.4 Educational laboratories

"Laboratories are places where elegant theories meet messy everyday reality" [59]. For engineering education, laboratories bridge the knowledge that is clearly printed in textbooks and the skills that can only be acquired through solving real-world problems. With the emerging of online distance education, educators began to search for suitable methods of delivering educational

[2] Tangling refers to parts getting looped together, for instance, two split rings may loop over each other. Nesting refers to parts getting stuck inside of one another, for instance, foam cups may get stuck when they are put in a pile. Both tangling and nesting may happen during part storage or shipping in bulk. Because excessive efforts are needed in order to separate tangled or nested parts, part designers should avoid features that causing tangling or nesting [100].

laboratories. In this document, as a VE for constructing remote educational assembly laboratories is described, it is necessary to review some existing or proposed remote educational laboratories.

For distance engineering education, can we deliver a “messy reality”, which always hides the truth with noise, errors and mistakes, through the internet? While there were doubts whether technically engineering educational experiments could be delivered remotely [60], there were also many solutions. Some of these solutions use computers to simulate the processes and results of experiments [61] [62]. Some other solutions, taking advantage of remote sensing and control technology, allow students remotely connect to a real experiment device [63] [64]. There are also some solutions that combine the virtual laboratories and remotely-controlled laboratories to create hybrid laboratories. [65] Through these laboratories, students are able to conduct pedagogical experiments under pre-programmed guidelines. However, these laboratories are short of flexibility as they only allow students to do pre-designed activities.

Most remote laboratories are web-based: students use internet browsers to interact with the laboratory system. They can use web-browsers to send commands and give inputs [66] [67]. They can also watch experiment processes, which may be simulated by 2D/3D computer graphics [68] [69], or captured by web camera at a remote location [70] [71] on web-browsers. Finally, results of experiments are presented to them, usually in the forms of lists or plots [72] [73]. These laboratories are easy for students to get started: the operations in these laboratories are similar to web-surfing and thus students do not need training to complete these tasks. However, such web-based laboratories fall short of providing a “messy reality”. For one thing, they do not provide feeling of immersion to students: experiments in real world are not done by clicking hyperlinks or typing in text box; for another, the presence of teamwork, which is crucial in real laboratories, is rarely integrated into such web-based laboratories.

VE technology can provide the feeling of immersion and support team collaboration [74] [75]; and state-of-the-art videogames engines can offer an inexpensive desktop virtual reality authoring workbench. Since 2007, a videogame-based laboratory platform has been developing by the author's research group [76]. The platform is constructed on GMod, a 3D first person shooting (FPS) game, as the VE authoring tool. Like all FPS games, during experiments, each student controls an avatar by mouse and keyboard. Through manipulating the avatar's movement, the student is able to mimic real-world experiment activities. In addition, avatars controlled by different students can share a same virtual laboratory room, thus they can "meet" and collaborate. In addition, unlike most VE authoring tools, which require extensive hardware investment, GMod can run on personal computers, by which the versatility of the platform is enhanced and its development cost is reduced.

2.5 Using Microsoft Kinect as input device

Although traditional input devices (keyboard and mouse) have been proven to be suitable for VEs, new input devices that have emerged in recent years can provide an enhanced immersion for VE users. The Microsoft Kinect is one of them and the author of this document has utilized it.

Using the Kinect for other purposes other than entertainment has become a popular topic since its introduction in 2011, when it was only sold together with the Xbox for entertainment purposes. In 2012, Microsoft released the Kinect for Windows with an open-source Kinect software development kit (Kinect SDK), thus allowing third parties to develop their own applications with less technical difficulty.

Its real-time gesture tracking function makes the Kinect an ideal tool for medical therapy and physical rehabilitation. For stroke recovery, research indicated that patients recover better with the help of the Kinect than using traditional methods [77] [78] [79]. Evaluations also showed that the Kinect can greatly improve the effectiveness of the rehabilitation for motor disabilities in young adults or children, because it can boost the patients' motivation [80] [81]. There are also reports on adapting the Kinect for other medical purposes such as senior-people monitoring [82] [83]**,** Parkinson's disease rehabilitation [84] [85], burn injury recovery [86], etc. Most of these medical applications employed VE technology or gaming environments to maximize the therapeutic effects.

The Kinect can potentially serve as a lecture tool. During lectures, it would be easier for instructors to use their gestures to control computers rather than to click a mouse or tap a keyboard again and again. For instance, an instructor can use a combination of gestures to control a slide show [87] [88]. Similarly, when instructors want to demonstrate the kinematics of a robot in a robotics class with the help of the Kinect, they can guide the robot by waving their arms, a method that is faster than operating the robot via a traditional interface and thus saves lecture time [89].

The Kinect can also be applied to strengthen student learning by facilitating kinesthetic interactions. For instance, research has shown that Kinect-assisted reading helps youths remembering more words [90]. When being combined with a VE, the Kinect can also improve the social competencies and executive functions of students with Asperger syndrome [91]. For autistic children, playing Kinect-based games may benefit their skill learning [92].

Furthermore, the Kinect can also serve as an intelligent student supervision tool. For instance, the Kinect's body tracking function was used to monitor the gestures of ballet students during practice [93]. In addition, the Kinect was combined with augmented reality technology to help young performing artists to improve their stage performance [94].

Chapter 3 Virtual assembly in a game-based VE

3.1 Comparison between VE and computer-aided design

From the functionality perspective, a game-based virtual assembly environment should have the following characteristics:

1. Flexibility in assembling: Parts must be allowed to connect if they have matching features and do not violate the assembly sequence.
2. Ease of operation: Compared with traditional CAD software, the immersive environment must provide a simple method for manipulating different parts.
3. Cooperation among users: Multiple users must be enabled to collaborate on the same assembly simultaneously (and these users may interfere with each other in positive or negative ways).

However, as game engines are designed for entertainment purposes, they focus on the simplification of simulations as a compromise to the limited computation power of today's computers. For example, the physical bounding volumes of most geometries are simplified to primitives such as cubes or spheres in order to accelerate the collision detection process. For the purposes of assembly simulations in CAD systems, the requirements are quite different. The simulation time is generally not as important as the accuracy. The operation of CAD software is also hard to master, especially for beginners. This is because the operations provided must be able to encompass all details during design.

For a game-engine based VE for mechanical assembly, two fundamental problems have to be solved. One problem is the topological modeling of mechanical assemblies, and the other is the simulation of assembly processes. In this chapter, a feature-based assembly representation for a game-engine based VE is discussed.

3.2 Assembly representation: set, relation matrix and undirected graph

An assembly is a set that encompasses a set of parts in conjunction with a set of relationships between these parts. Therefore, an assembly can be represented as:

$$A = \{P, R\}$$

In the equation, P is the set of parts and R is the set of relationships.

P is a set that includes all parts:

$$P = \{P_1, P_2, \ldots, P_N\}$$

If multiple parts of the same type appear in an assembly, each of them should be noted separately in this set.

The relationship set R consists of 2-tuples of parts:

$$R = \{< P_1, P_2 >, < P_1, P_3 >, \ldots, < P_K, P_N >\}$$

Two types of data structures can explicitly describe an assembly. One is a relationship matrix and the other is an undirected graph of connections. A relationship matrix is a square array with rows and columns representing the parts. The matrix contains '0' and '1' values only. A value of '0' means that two parts have no direct connection and a value of '1' indicates that two parts do have a direct connection. A relationship matrix can be represented by an undirected graph with nodes and edges. A node in the graph denotes a part, and an edge between two nodes indicates that these two parts have a direct connection.

For instance, Figure 1 shows a bolted joint that consists of a threaded bolt, two plates, a washer and a threaded nut. Several connections are included: the bolt and nut are connected by threads, the plates and washer are centered by the bolt, and these components are axially fixed to each other. Therefore, the part set P is:

$$P = \{B, P_1, P_2, W, N\} \tag{1}$$

Here, B denotes bolt, P_1 denotes plate 1, P_2 denotes plate 2, W denotes washer, and N denotes nut.

From Figure 1, it can be seen that the bolt is the central piece in the sense that all other parts have a direct connection with it. In addition, the two plates, the washer and the nut only have connections with their respective neighbors. Therefore, the relationship set R is:

$$R = \{< B, N >, < B, P_1 >, < B, P_2 >, < B, W >, < P_1, P_2 >, < P_2, W >, < W, N >\} \tag{2}$$

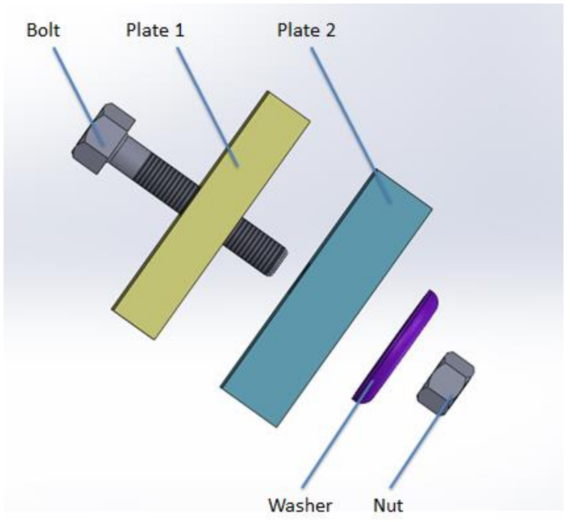


Figure 1: Exploded view of a bolted joint

The assembly can therefore be described by an adjacency matrix (AM):

$$AM = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

Here, the columns and rows represent parts *B*, *P*$_1$, *P*$_2$, *W and N*, respectively. The digit "1" means that an inter-part connection exists while digit "0" indicates that there is no such connection.

It can be seen that the adjacency matrix is always symmetric and the main diagonal elements are all equal to zero. (They can be assigned another number such as "-1" as one part cannot connect with itself in any case.)

The advantage of the adjacency matrix is that it is easy to document; its disadvantage is that it is hard to modify. If a new part is needed, a new row and column must be inserted into this matrix. Furthermore, this matrix can be very sparse if there are many parts in an assembly, considering that usually a part has only few direct connections with other parts.

Figure 2 shows the undirected graph of the bolted joint depicted in Figure 1. There are 5 nodes in the graph, representing the 5 parts of the bolted joint. The edges between these nodes represent the direct connections between these parts.

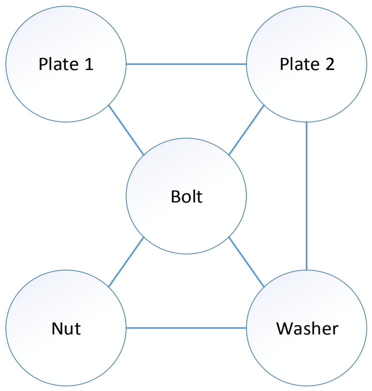


Figure 2: Undirected graph of bolt-nut connection

However, in both representations, a relationship merely reflects a ‘connection’ of two parts without any additional details. In order to provide more details of this ‘connection’, the part as well as its features and feature associations should be defined.

3.3 Parts, features, feature associations and assemblies

3.3.1 Feature identification on parts

In an assembly within the immersive VE, a part is a fundamental functional element which cannot be divided any further in order to maintain its functionality. In this proposal, a part is assumed to be a rigid object.

Because of this assumption, a part can be stored in the computer as a single 3D model. Although multiple model representations exist, with some specifically designed for mechanical modeling, in the game-engine powered immersive VE proposed here, a part can only be stored by its triangular boundary representation, which consists of positions of vertices and triangular meshes defined by these vertices. Any form features are therefore absent from this model.

However, as previously stated, form features are now prevalent in modern computer-aided design. Engineering designers would like to represent a part as a set of form features instead of triangles. Therefore, features of a part should be identified. Although automatic feature identification algorithms have been reported [95] [96], in this proposal, feature identification is performed manually because for assembly, only assembly-related features are needed while most automatic feature identification algorithms would identify all form features on a part with uncertain accuracy.

Figure 3 depicts an assembly of a stepped shaft with two bearings, a threaded tightening, a tightening ring and a fan mounted on it. Figure 4 shows the features defined and identified on this shaft. Among many cylinder features associated with the stepped shaft, threaded cylinder 1, cylinder 2, cylinder 3 and parallel key features are identified for assembly (labeled by normal font in the figure). Other form features, such as fillet, cylinder 5, cylinder 6 and cylinder 7 are not identified as they are not designed to connect to anything and thus are irrelevant for the assembly (labeled by italic font in the figure).

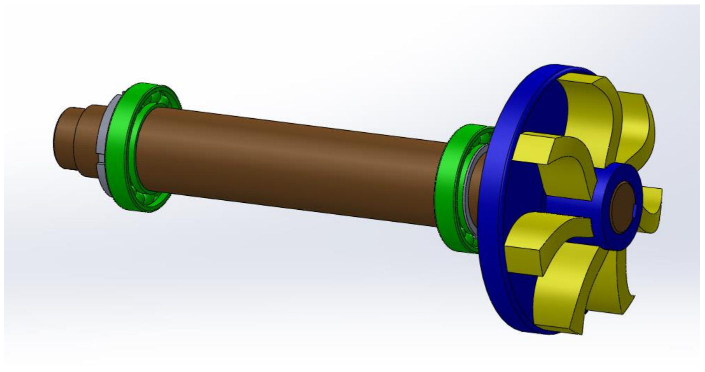


Figure 3: Assembly of a stepped shaft with a fan

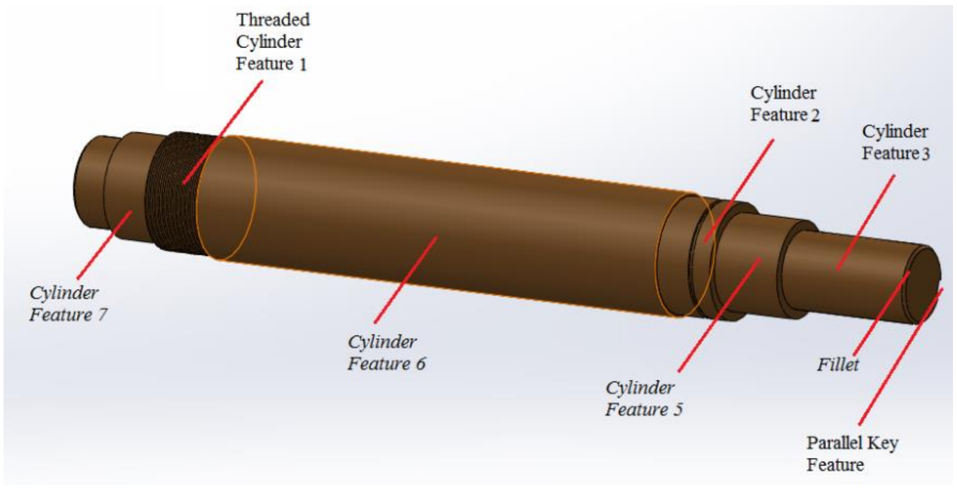


Figure 4: Identification of assembly features

3.3.2 Feature slots

In a slot-modular product architecture, a slot is a standardized interface between two functional components. In contrast to a bus-modular architecture, in the slot-modular architecture, within a product, the slots between different components are varied and non-interchangeable [97]. The concept of feature slot is similar. However, feature slots are interfaces of feature-based parts, rather than of functional components.

A feature slot contains not only the feature information but also other attributes (see Table 1).

Table 1: Major attributes of a feature slot

Attribute Name	**Description**	**Example**
Feature_type	Type of feature	Cylinder feature; notch feature
Feature_position	Position of feature in part's local coordinate system	(x_1, y_1, z_1)
Feature_orientation	Orientation of feature in part's local coordinate system (not applicable to some features, for instance, spherical feature for ball joint)	(x_1, y_1, z_1)
Feature_parameter	Specific parameter(s) to describe dimensions of feature	Diameter; depth
Feature_availability	Flag for tracking whether feature slot has connected with another feature slot via feature association	'True' or 'False'

Among these attributes, *Feature_position* and *Feature_orientation* define the placement of a feature on a part. They are stored as a 3D vector that refers to the local coordinates of that part, which is associated with that part when it is modeled. No matter how this part is translated or rotated, the values of these two attributes are not changed.

In addition, the position of a feature slot is not necessarily fixed. In most cases, it does not move, though. For instance, the diameter of the shaft end matches that of the corresponding bearing. In some other cases, the position of the feature may be flexible within a certain range (e.g., a slider). The slider can be assembled at any place of the guide rail. In this case, the operator is asked to assign the position of the feature slot. Furthermore, the position may change due to some motion of the slider.

3.3.3 Assembly feature associations

An assembly is created by connecting two or more parts. Some parts are supported by other parts. In the assembly in the immersive VE, such connections are implemented by assembly feature associations. When connecting two parts, the matching assembly feature is associated according to the feature association type. For instance, as shown in Figure 5, a shaft has a key slot feature, a cylinder feature and a plane feature. Analogously, a gear has a key slot feature, a cylindrical hole feature and a plane feature, and a key has a key feature. These form features jointly determine a shaft-key-gear assembly. In detail, these feature associations are:

1. Cylinder - cylindrical hole association
2. Key - shaft key slot feature association
3. Key - gear key slot feature association
4. Gear side plane - shaft shoulder plane coplanar association

In the game-based VE, the number of associations in an assembly could be reduced by merging associations. For instance, in the mechanical design depicted in Figure 5, in order to prevent a gear from moving in the axial direction, the shoulder on the shaft and the side of the gear form a coplanar association. However, in some cases when a VE author wants to simplify the assembly, he/she can define that the cylinder-cylindrical hole association additionally prevents axial movement. By doing this, the coplanar association can be eliminated from the association list.

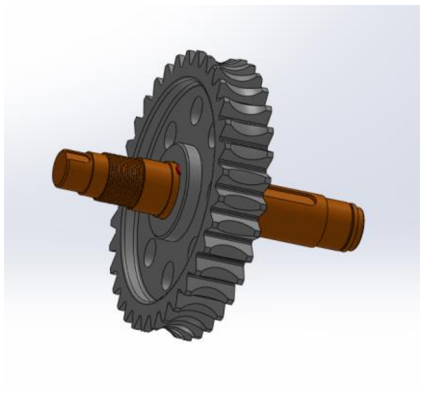


Figure 5: Shaft-gear-key feature association

3.3.4 Kinematic feature association

Assembly feature associations enforce connections between parts to form sub-assemblies. That is, when one part is translated or rotated, the associated part must translate or rotate accordingly so that the kinematic constraints between these parts are not changed. Contrary to this, kinematic feature associations do not form sub-assemblies. There is no bonding force on the joint to ensure that a mechanical assembly is made. For instance, a gear-meshing association imposes a kinematic constraint between two gears only when they are meshed. When one gear is moved away, the association no longer imposes any constraint.

3.3.5 Assemblies and sub-assemblies

After the feature associations have been clarified, defining the concepts of sub-assemblies and assemblies becomes straightforward. A sub-assembly is made up of some parts that are associated by assembly features. Similarly, an assembly is made up of several sub-assemblies. The connections between sub-assemblies are also implemented by feature matching on matching parts. It should be noted that features are still on parts and sub-assemblies themselves do not have any features.

3.4 Assembly topology with features

In the game-engine powered immersive VE proposed here, an assembly is stored in a graph as shown in Figure 6. Each node represents a part, and each part has one or multiple features. As stated previously, parts can be connected by both assembly and kinematic feature associations.

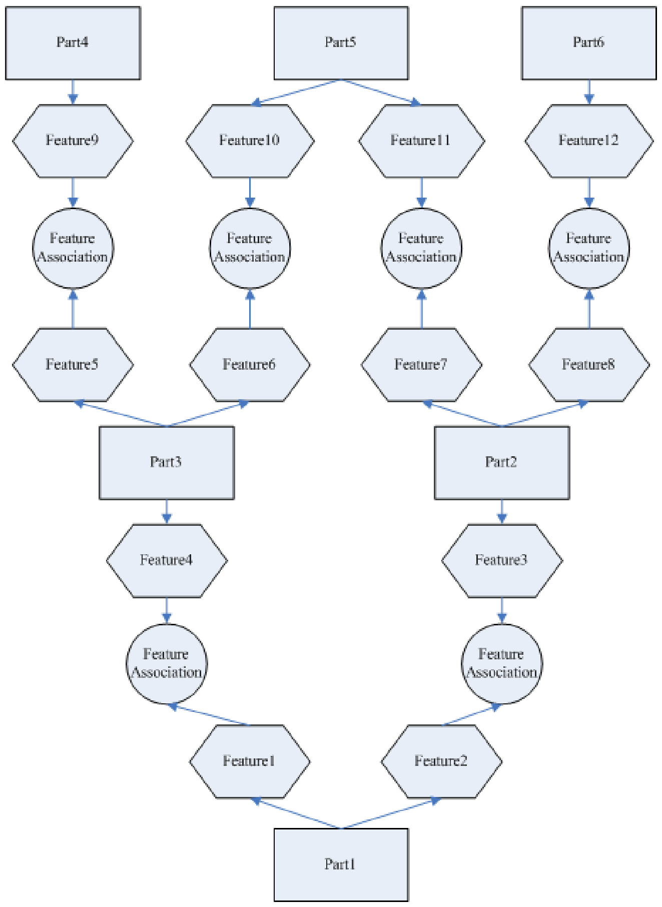


Figure 6: A tree structure to represent an assembly

In most cases, an assembly graph represents a tree structure such as the one depicted in Figure 6. That is, one node can only have one parent node, but one parent node can have multiple child nodes. For instance, in a shopping cart assembly, casters can be assigned as child nodes as they only have associations with a cart body while the cart body is a parent node, which is associated with not only casters but other accessories, such as baby seat, handle, etc.

The assembly graph can also be a loop. For instance, in Figure 7, four parts are connected by cylinder-cylindrical hole feature associations. On each plate, there are 2 cylindrical hole features, and on each cylinder-shaped part, there are two cylinder features that can be associated with the holes on the plates.

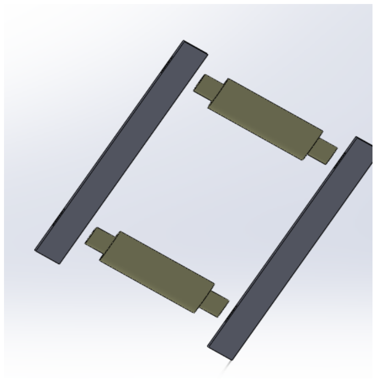


Figure 7: An assembly with a loop

3.5 Hierarchy of assemblies

3.5.1 Assigning level ranks to parts

Although all edges of an assembly graph are bi-directional, from the perspective of mechanical assembly, the establishment of assembly associations does have a direction. In order to make an assembly, one always moves a part or sub-assembly to another part or sub-assembly. For instance, when installing a nut on a bolt that has been fixed to a part (i.e., screw and part form a sub-assembly), thus establishing a fixed association (i.e., removing all 6 relative DOFs), one usually manually moves the nut to align with the bolt rather than moving the whole sub-assembly to align it with the nut. In order to represent this assembly association direction, a hierarchical structure is used to describe an assembly. Each part in the assembly is assigned a rank in the hierarchy.

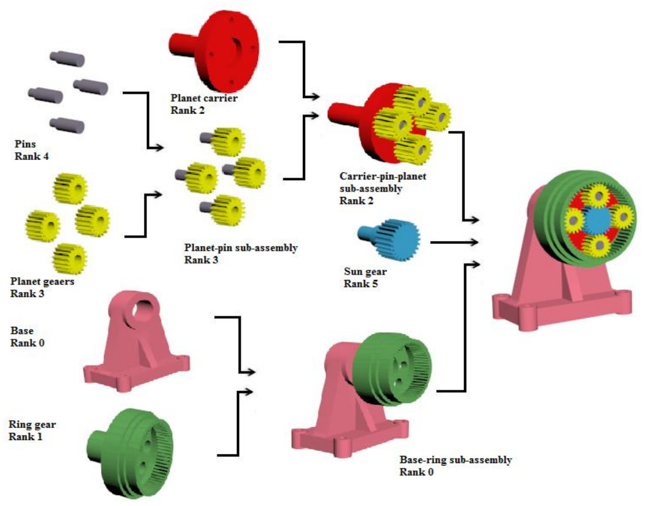


Figure 8: Demonstration of an assembly hierarchy

For kinematic feature associations, such as a gear meshing association, the direction is not necessary, since associations of this type do not create sub-assemblies. Although there are functional differences between driving and driven gears, in order to let two gears engage, one does not have to move the driving gear to the driven gear or vice versa.

3.5.2 Rules for assigning ranks to parts

When assigning the ranks to the parts of an assembly, the basic principle is to assign the highest rank to the foundation part, which supports the whole assembly. The lowest rank is given to decorative parts or parts that do not support any other parts. Therefore, in most cases the rank is related to the sequence of the assembly. In Figure 8, the base part represents the foundation of the whole gear train as all other parts are mounted onto it. Hence, the base has the highest rank, namely rank 0. The ring gear is directly linked with the base by a revolute joint, and it thus has the second highest rank, namely rank 1. When the ring gear is ordered to be associated with the base, the virtual system automatically decides to move the ring gear to the base rather than the other way around. The planet carrier, as an intermediate part, is ranked 2, lower than the ring gear and higher than the other parts, since on one hand, it is attached to the ring gear, and on the other hand, it holds all planet gears and the sun gear.

It appears that the rank assignment is related to the assembly sequence but this is not always true, as there may exist two or more allowed assembly sequences. Hence, there is a possibility that a rank that is suitable for one assembly sequence will not allow another assembly sequence. For example, the preferred sequence of assembling the planet gear and pin is to first insert the pins into the carrier and then to mount the planet gears onto the pins. Therefore, since the rank of the carrier is 2, the rank of the pins should be 3 and that of the planet gears should be 4. However, it is also acceptable to connect the planet gears and pins first to create sub-assemblies and then to insert the sub-assemblies into the holes in the planet carrier. In this situation, since the pins are ranked 3 and the planet gears are ranked 4, during the formation of the sub-assembly, the planet gears move to the pins rather than the pins being inserted into the holes of the planet gears. In order to better fit this assembly sequence, the rank of the pins should be 4 and that of the planets should be 3. However, this rank assignment would then not be suitable for the first assembly sequence.

In order to resolve this issue, the concept of the rank of sub-assemblies was defined. When two parts form a sub-assembly, the rank of the sub-assembly is assigned to be the rank of its highest-ranked part. When another part or sub-assembly is mounted onto this sub-assembly, the rank of the sub-assembly is compared rather than the rank of the connecting part. In the gear train example, the pins and planet gears are assigned rank 4 and rank 3, respectively. When a pin is first inserted into the carrier, then the rank of the carrier-pin sub-assembly takes on the rank of the carrier, which is 2. Since the planet gear is ranked 4 and the sub-assembly is ranked 2, when mounting the planet gear onto the sub-assembly, the planet gear is the lower-ranked part and moves to the fixed pin. If the other assembly sequence is applied wherein a sub-assembly of pin and planet gear is created first, then the pin moves to the planet gear since it is the lower-ranked part to form the sub-assembly. The sub-assembly inherits the rank 3 of the planet gear. Then, in the process of assembling the sub-assembly with the planet carrier, because the rank of the sub-assembly is still lower than that of the planet carrier, the latter would be fixed and the sub-assembly would move to the planet carrier, as it would happen in the real world.

In an assembly, two different types of parts that are not associated may share the same rank, although this is not recommended. In the planet gear train, the sun gear can have the same rank as the planet gears without affecting the assembly process.

3.6 Assembly sequence constraints

3.6.1 Difficulties caused by bounding volume

In most 3D games, collisions are simulated. In order to determine whether two objects collide, game engines periodically run a collision detection process, which checks the position of all objects and seeks any overlaps between two objects. This process consumes a lot of computational power because on one hand, in order to deliver a fluent game play, the period must be very small (usually less than 0.1 s), but on the other hand, finding overlapping volumes are complicated. Therefore, collision detection algorithms for games rely on approximations. There are many methods of approximation, most of which include attaching a ‘bounding volume’ (i.e., a simplified geometry) to the actual complex geometry mesh. Although this method decreases the precision of collision detection, the collision detection efficiency is increased as the number of surfaces on each object is decreased dramatically.

The lower precision does not affect the game playing too much, but it is detrimental for the mechanical assembly VE. An example is shown in Figure 9. For a gear, the involute-shaped teeth are represented by tens or even hundreds of small surfaces. It would be unrealistic for collision detection to be applied to this model directly. Therefore, game engines assign a bounding volume to such complex shapes. In most simple cases, the bounding volume is a cylinder. Here, this cylinder has a diameter that is slightly larger than the addendum diameter of the gear.

Such an approximation causes a problem for assembly processes. When a simple gear train is designed, the dimensions and positions of input and output gears are determined first. Based on the input and output gears, the dimension(s) and position(s) of (an) idler gear(s) can be determined. Then, the idler gear(s) can be inserted between these two gears. However, based on the bounding volume, this insertion is invalid, because the space left between the input and output gears' bounding cylinders is smaller than the bounding cylinder of the idler gear. In addition, the dynamics of involute gear tooth meshing cannot be simulated using such a method of approximation. Therefore, it is necessary to define rules that supersede the bounding volumes for assemblies in some situations.

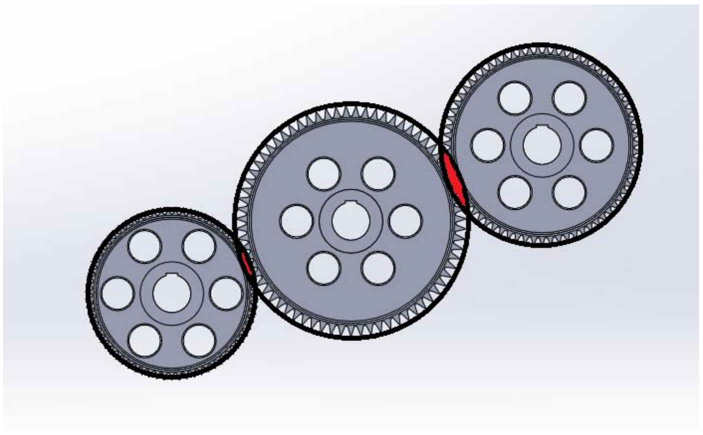


Figure 9: Bounding volumes of a gear train

3.6.2 Algebraic representation for assembly sequence constraints

One way to solve this problem is to disable the bounding volumes and use the gears' actual geometries instead. In this case, the positions and diameters of the gears are used to determine if a particular assembly is possible. Before constructing kinematic feature associations of gear engagement, the following rule of diameters must be checked. Assuming that all gears are coplanar, in order to guarantee that gears A and B are engaged, the assembly must satisfy:

$$\boldsymbol{p_A} + \boldsymbol{r_{A/AB}} + \boldsymbol{r_{B/AB}} = \boldsymbol{p_B} \qquad (4)$$

Similarly, in order to ensure that gears B and C are engaged, the assembly must satisfy:

$$\boldsymbol{p_B} + \boldsymbol{r_{B/BC}} + \boldsymbol{r_{C/BC}} = \boldsymbol{p_C} \tag{5}$$

In the first equation, $\boldsymbol{p_A}$ is the 2D position vector of the center of the gear, $\boldsymbol{r_{A/AB}}$ is the radius vector of the gear, the length of which is the pitch radius of gear A, and the direction of which is from the center of gear A to the center of gear B.

3.6.3 Logic representation for assembly sequence constraints

If one establishes an assembly following a wrong order, the most likely result is that one part blocks the assembly of another part. However, as the collision detection for assembled parts are superseded by the rules described above, the blockage may not be enforced automatically anymore by the VE. A new representation of blockage is therefore devised.

For instance, as shown in Figure 10, in order to assemble a shaft that carriers a gear, one must first assemble one bearing with the shaft, then mount the gear on the shaft in the next step, and finally assemble the other bearing. If the second bearing is mistakenly assembled without mounting the gear first, then the gear cannot be assembled anymore. A logic equation is applied to rule the blockage.

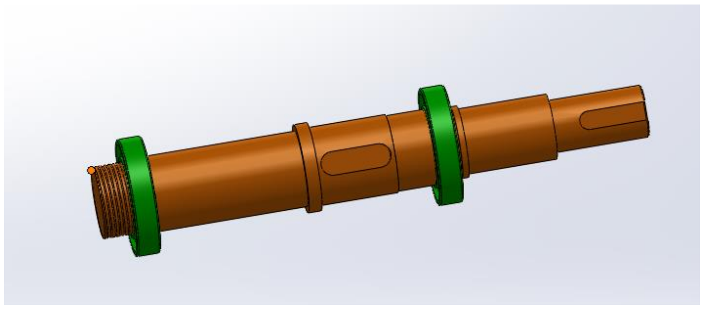


Figure 10: A shaft with two bearings

In the problem depicted in Figure 10, the logic equation can be written as follows:

$$A(Gear) = \neg(O(LB) \land O(RB)) \tag{6}$$

where *A(Gear)* is the availability of the gear slot and *O(LB)* and *O(RB)* is the occupancy status of the features of the left and right bearings, respectively.

Once the left bearing has been connected to the shaft, the Boolean value of *O(LB)* is true, and so is *O(RB)*. If both of them are true, the value of *A(Gear)* is false, meaning that the gear cannot be mounted anymore. If either *O(LB)* or *O(RB)* is false, i.e. one of the bearings has not been connected yet, *A(Gear)* is true and the gear can be mounted on the shaft.

A major drawback of these assembly rules is problem-specific. For some assemblies, the algebraic representation helps while for some other assemblies, the logic representation is easier. Furthermore, for each assembly the equations for sequence rules differ.

3.7 Creating assemblies in the game-based VE

3.7.1 General procedures for creating assemblies

After defining the assembly representation concepts of assembly feature, assembly feature association and assembly rules, the following question emerges: Can one let people who use this VE conveniently create assemblies?

In CAD software, designers can select, usually by mouse, exact mating surfaces to create assemblies. However, in real world scenarios, this is unrealistic. People tend to not really touch these mating surfaces. Instead, people align parts and conduct assembly operations such as insertion etc. Therefore, in the VE, a similar procedure for creating assemblies was developed. One can control an avatar in the VE to grab parts, move and align them, and then assemble them. This alignment may not be precise due to the input interface of the VE. Therefore, the VE needs to adjust the assembly position using a method described in Section 3.7.2.

In real-world scenarios, the geometry of assembly features determines whether two parts can be assembled and what type of feature association there is. The VE should also be able to make such decisions. Therefore, after two parts have been aligned, the VE performs the following 3-step algorithm:

1. Identify a pair of matching feature slots
2. Determine the exact assembly position
3. Establish corresponding feature association and impose kinematic constraints between two parts

3.7.2 Identification of matching pairs of assembly slots

When one controls an avatar such as to align any two parts and cause them to collide, the VE initiates a process of determining whether there are any pairs of feature slots that can be associated. The flow chart of this process is shown in Figure 11.

After identifying all feature slots on both parts, the first test is to remove all feature slots that have been associated with other parts. While one part can be associated with multiple parts, a feature slot can only be associated with one part. Then, the VE determines whether there are available feature slots to connect. As previously introduced in Section 3.3.2, a feature slot is a class that stores the information of a feature on a part. When two parts are to be connected, they must have matching features. Finally, the VE further examines if these type-matching feature slots have matching parameters (e.g. diameter, depth, etc.).

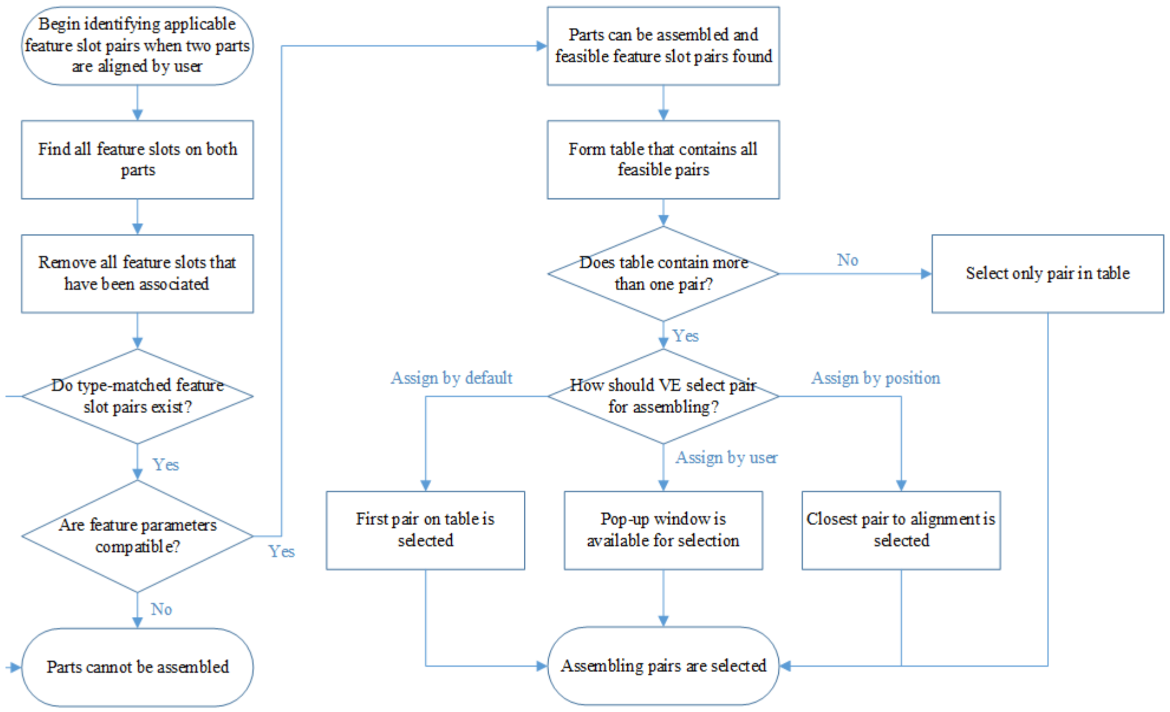

Begin identifying applicable feature slot pairs when two parts are aligned by user
Find all feature slots on both parts
Remove all feature slots that have been associated
Do type-matched feature slot pairs exist?
Yes
Are feature parameters compatible?
Yes
No
Parts cannot be assembled
Parts can be assembled and feasible feature slot pairs found
Form table that contains all feasible pairs
Does table contain more than one pair?
No
Select only pair in table
Yes
How should VE select pair for assembling?
Assign by default
Assign by position
Assign by user
First pair on table is selected
Pop-up window is available for selection
Closest pair to alignment is selected
Assembling pairs are selected

Figure 11: Identification of qualified pairs of assembly slots

It is possible that there is more than one pair suitable for making an association. However, for an assembly between two parts, only one pair of matching slots can be applied for establishing an association. Based on the feature type, the VE can automatically use the pairs that have the smallest feature slot number, which is suitable for parts that have an array of identical feature slots. Alternatively, the VE can ask the user to select which pair should be used, which helps the user to avoid errors, or it can assign the pair that is closest to the alignment position, which may reflect the user's intention without interrupting the assembly process. After these processes, a matching pair of assembly slots is identified by the VE.

3.7.3 Determination of exact assembly position in world frame

After the matching pair has been assigned, the part in the lower rank is directed to its assembly position in the global coordinate system by the VE.

As previously stated, in a part, the positions of all feature slots are defined in a local coordinate system attached to the part. The transformation from the local frame to the world frame includes four steps:

1. Get position and orientation of matching feature slot on lower rank part by its current position and orientation in world frame
2. Transform position and orientation of higher rank part into world frame and compute position and orientation of feature slot for association
3. Based on position and orientation of feature slot of higher rank part in world frame, calculate position and orientation of this part in world frame
4. Disable collision detection among all assembled parts

An example of the coordinate systems is shown in Figure 12. There is a carrier at a higher rank and a pin at a lower rank. For the carrier, its local frame is attached at its center, with the z-axis aligned with the shaft axis. Its position is at (X_1, Y_1, Z_1) in the global frame and its orientation is aligned with the global frame. Therefore, for a hole feature slot (feature #1 on the carrier) located at (x, 0, 0) in the local frame, its world frame coordinate is (x+X_1, Y_1, Z_1) and its orientation is (0, 0, 0). In order to insert the pin, the cylinder feature slot on the pin must be in the same world frame position and orientation as the hole feature slot on the carrier. Since the pin is a lower ranked part, the pin must be moved to the carrier. Regardless of what was the coordinate of the pin in global frame at the beginning, the pin must be relocated to the world frame position (x+X_1, Y_1, Z_1) and orientation (0, 0, 0) in order to facilitate the connection of the cylinder feature slot (feature #2 on the pin) to the hole feature slot (feature #1).

If the part in the lower rank is connected to another part (i.e. is a part of a sub-assembly), the VE must not only compute the position and orientation of the lower part in the world frame, but also those of all sub-assembly parts.

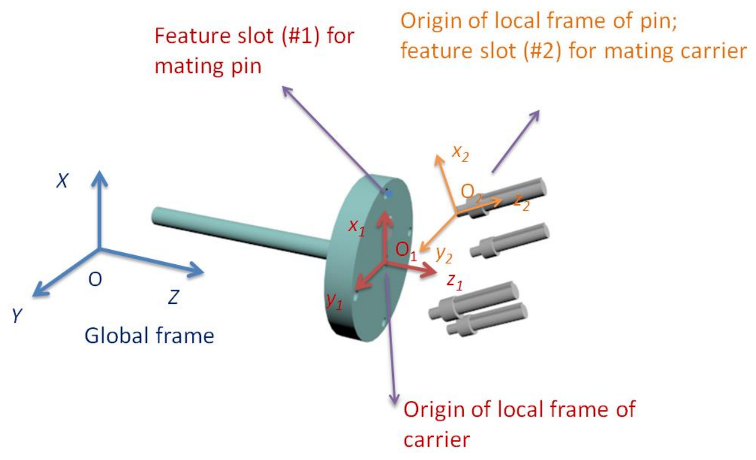

Feature slot (#1) for mating pin
Origin of local frame of pin; feature slot (#2) for mating carrier
x_2
O_2
z_2
y_2
X
x_1
O_1
z_1
y_1
O
Z
Y
Global frame
Origin of local frame of carrier

Figure 12: Coordinates for carrier and pins

3.7.4 Establishment of feature associations

After the part in the higher rank has been positioned to the correct place and orientation, a feature association can be established between these two parts. The feature association connects two parts by removing a certain number of relative DOFs.

Most VE engines provide a library of associations for VE authors. For instance, in GMod, a VE authoring tool introduced in the next chapter, these associations are defined, and they will be referred to as ‘constraints’. Some commonly used feature associations are listed in Table 2.

Table 2: List of feature associations

Constraint in Game-engine (GMod)	Feature association in mechanical assembly	Description	Number of DOF removed
Axis constraint	Revolute joint	Joins two parts with an axis about which they can spin freely; their relative positions in axial direction are fixed	5
Ball socket	Spherical joint	Joins two parts with same center point about which they can rotate freely in all directions	3
Elastic constraint	Elastic joint	Connects two parts with a spring-like rope that, when compressed or stretched, tries to resume its original length	5
Slider	Prismatic joint	Creates a path along a straight line on a part that a matching part can travel along	5
Weld	Fixed joint	Joins two parts such that afterwards they can no longer be moved relative to each other	6

3.7.5 Disabling of collision detection within a sub-assembly

As introduced in Section 3.6.1, collision detection algorithms may cause problems in simulating assembly processes. However, the VE needs collision detection, for instance to prevent parts from penetrating a wall. Therefore, collision detection should be disabled only between connected mechanical parts.

When an association between two parts is established, not only the collision detection between these two parts is disabled, but also for all parts that have formed sub-assemblies with them (if any).

3.8 Chapter summary

In this chapter, the assembly essentials were described. The bi-directional assembly graph is the core part of an assembly. In this graph, two connected parts are linked by a feature association, which contains two mating feature slots. A feature slot is a set that contains feature type, feature parameter and other feature-related information.

In order to represent the sequence of an assembly in a flexible way, a hierarchical model for assemblies was introduced with each part in the assembly being assigned a rank. The assembly order is also constrained by Boolean or algebraic equations.

The VE needs to validate an assembly activity before generating any assembly. The VE checks whether there are available feature slots for connecting and furthermore checks whether the parts have collided in the correct position and orientation. Once an assembly of two parts has been validated, the algorithm either assigns a feature slot on each part or asks for the user to assign appropriate feature slots. Once these feature slots have been assigned, their corresponding feature association is generated automatically. In addition, any parts that are not directly related are linked by 'no collision' constraints.

Chapter 4 Using GMod as VE authoring tool

4.1 Why VEs instead of CAD software

Concepts of CAD systems can be adopted for the implementation of game engine based VEs for simulating assembly processes. Compared with traditional CAD environments, a game-engine powered immersive system largely facilitates the design review and assembly training. The advantages of a VE system include:

1. Face-to-face experience: The participants in an assembly process are displayed in the environment as human-like avatars who can talk to each other either by text messages or by voice chatting.
2. Near-real scenarios: The VE is capable of providing simulated assembly places. Also, events can be added to enhance the immersion. For instance, a sudden machine breakdown can be simulated in order to test the stability of a designed assembly process.
3. Support of group activities: An immersive assembly VE allows the simulation of multi-player collaboration. Therefore, the participants are enabled to work on the same task while playing different roles.
4. Activity tracking: Assembly activities by all participants can be tracked and recorded by the immersive VE.

In this research, an implementation of the proposed immersive VE is included. A multi-player laboratory environment for students to learn about gear mechanisms was designed and tested. This environment is based on GMod, a modification of the Half-life game engine. This educational laboratory utilizes the assembly representation for immersive VEs as described in Chapter 3.

In the first part of this chapter, an assembly of a planetary gear train is introduced. The parts, features, feature associations and assembly procedures for the planetary gear train are illustrated. In the second part of this chapter, the educational laboratory for undergraduate students to learn about planetary gear mechanisms is demonstrated and the laboratory process, requirements, questionnaires, tutorials and the students' feedback are discussed.

4.2 Requirements for VE-based laboratories

A game engine is a platform that supports the development of games. With the help of a game engine, one or more games can be developed with similar styles but quite different contents. For instance, Half-life series games and Counter-strike series games are both developed based on the 'Source' game engine. From a content point of view, Half-life games provide full story lines for players to challenge while Counter-Strike games are team-based freestyle games. However, from the point of view of gaming factors (such as graphics, physics simulation, audio, user-interfaces, etc.), they are the same.

In order to support different games, the factors that game engines address vary. For instance, for the engine that supports Half-life 2, graphics effects were made a priority while AI was not as important. Contrarily, for the engine that supports Civilization, AI is the key factor. Therefore, in order to select a game engine for the implementation of a VE for mechanical laboratory exercises, the importance of the gaming factors should be analyzed. Table 3 lists the major gaming factors and their respective requirements and priorities for virtual mechanical laboratory exercises.

Table 3: Requirements and priorities of gaming factors for virtual mechanical laboratory exercises

Factor	*Requirement*	*Priority*
Graphics	Engine should be able to render geometries in high fidelity.	High
Audio	Only basic sound effects are needed.	High
Physics	Real-time simulation of real world physics and efficient collision detection are required	Very high
Network	Participants should be able to see each other in VE and communicate with each other by text or voice.	Medium
Artificial Intelligence	No artificial intelligence is required at this stage.	Very low

From Table 3, it can be seen that the physics engine has the highest priority since virtual mechanical laboratory exercises are targeted at strengthening the students' understanding of machines, the simulations of which are supported by the physics engine. A low quality physics engine, even when combined with the best graphics, would only confuse the students. However, the graphics engine and the sound engine are also important as they are the main factors that control the immersive feeling of the virtual laboratory. The virtual laboratory also requires that the network engine must include certain communication functionalities because interactivity is an important feature. In the virtual laboratory, the participants can see each other's avatars and talk to each other, which enables teamwork.

For automated tutoring systems, AI is vital. However, the implementation of automated tutoring needs extensive efforts. All instructions and help functions are given by human teaching assistants or instructors in the virtual laboratory. Therefore, AI was determined to be of least importance.

In order to satisfy these requirements, first-person-shooting game engines can be seen as the best choice, as most game engines of this type include graphics engine, audio engine, network module, AI engine and physics engine that cover all factors mentioned above. In a first-person-shooting game engine, the graphics engine is responsible for displaying and managing the data related to the graphical content and visual effects. Large numbers of geometries in the 3D VE, such as ambient environment, avatars, mechanical devices, etc. can be rendered in a real-time manner. Regarding shading, lighting, texturizing, particle effects, etc., these game engines can yield high fidelity effects that are above and beyond the requirements of the virtual laboratory. The audio engines in these game engines can also yield sufficient sound effects for the virtual laboratory. The network module in first-person-shooting game engines also satisfies the virtual laboratory's requirements as such attracting players by multi-user online gaming. Therefore, the avatars in the virtual laboratory are able to support 'face-to-face' collaboration under the same roof. AI engines were originally designed for the purpose of simulating the NPCs, but in the virtual laboratory, there are no NPCs at present.

However, the physics engine, which provides gaming physics and has the highest priority in the virtual laboratory, can only partially fulfill the needs of the virtual laboratory. On one hand, the physics engine does support basic physical effects such as collision, kinematics, forces (gravity, friction, contact, etc.). On the other hand, these effects are usually simplified and thus not accurate. For instance, models are usually bounded by a bounding box or cylinder in order to facilitate the collision detection between two models, no matter how complex the geometry of the models may be. In first-person-shooting games, such simplification is preferred as it saves computation power. However, for virtual laboratories, such simplification causes difficulties. For instance, hole-cylinder matching, which is common in mechanical assembly, cannot be simulated realistically using the physics engine, because firstly, the bounding volume of a model is usually larger than the actual part (i.e. the hole is smaller than it is designed to be while the cylinder is larger than designed), and secondly, even if the bounding volume of the hole and the cylinder could be modeled exactly as the designed size, a friction force between these two matching parts would be simulated and hence cause problems.

Unfortunately, there is no alternative with a better physics engine for virtual laboratories, especially considering the overall effects that first-person-shooting game engines provide. Fortunately, some physics effects can be overwritten by virtual laboratory developers while some others can be kept. For instance, depending on the context, the hole-cylinder matching can be alternatively modeled as a revolute joint or welding joint, and furthermore, the collision detection between the hole and the cylinder can be disabled so that the over-sized bounding box or friction will not affect the physical simulation regarding the matching.

In short, a first-person-shooting game engine is the best choice for the virtual laboratory, as it can satisfy the needs with respect to graphics, audio, networking and AI, while the shortcomings of the physics engine can be overcome.

4.3 Selection of GMod as platform

Although many functions are ready to use, scripting directly at the game engine level still requires a lot of work and a deep understanding of gaming technology. Thus, this approach is not feasible for inexperience software development teams.

An improved platform, GMod, was therefore applied for the work presented here. GMod is a Source-engine-based ‘physics sand box game’ which, instead of predefining any gaming scenarios, allows players to freely build their own contraptions, such as cars, rockets and catapults, using tools and models provided. As a game that is constructed based on the ‘Source’ game engine, on which first-person-shooting games such as Half-Life and Counter-Strike were based, GMod maintains the characteristics of first-person-shooting games.

However, a game itself is not suitable as a development platform. Luckily, beyond the gaming features, GMod allows game developers to import 3D models from third-party modeling software (e.g., 3ds Max) and construct their own game scenarios by ‘Lua script’, a computer language with a simple syntax. By these two development features, virtual mechanical laboratories can be implemented.

4.4 Preparation of mechanical models

As mentioned in Section 4.3, models can be imported from third-party modeling software. In GMod, mechanical parts can be treated as gaming models, which are stored in several files. These files record the properties of the model, such as model geometry (*.mdl file), model physics (*.phy file) and model level of detail (*.vtx file). In this section, the procedures of creating these files are presented.

Utilizing mechanical 3D CAD software (e.g. SolidWorks or Pro/Engineer), geometries of mechanical parts can be precisely and conveniently generated. These geometries can then be saved in STereoLithography (*.stl) format, which applies standard tessellation language to discretize the surfaces of the parts into triangles.

The next step is to texturize the models. In GMod, textures are pictures with a resolution of 32×32 pixels, 64×64 pixels, 128×128 pixels, 256×256 pixels or 512 ×512 pixels, which are stored in a Valve Texture File (*.vtf file). All models are required to be texturized by pictures in this file format in order to allow proper rendering in GMod. Texturizing can be implemented in 3D modeling software for art design (e.g., Autodesk 3ds Max). The advantage of this type of software, compared to mechanical CAD software, is that it provides flexibility of vertex manipulation and accuracy of texture mapping. Therefore, art design software is widely applied in modeling for gaming.

After texturizing, the model is exported to an ASCII-based StudioMDL data file (*.smd file), which not only includes the modeling triangles but also texture coordinates. If the model is to be animated, the corresponding animation information is also included in this file.

In GMod, collision detection and physics simulations cannot be directly applied to the geometry of a mechanical part, which is usually too complex to compute. Instead, they are applied to bounding volumes that encompass the geometry. A bounding volume can either be a single convex shape or a concave shape that consists of several convex shapes. Bounding volumes are applied because they can greatly simplify the computations for physical simulations without losing the realism. Bounding volumes can also be generated in 3D modeling software and then transferred into a *.smd file. The most commonly seen bounding volume is a bounding box.

The last step is compiling the .smd file into the gaming models that are stored in binary *.mdl, *.phy and *.vtx files. A specific compiler, ‘GUIStudioMDL’, is used to convert the ASCII-based *.smd file into the binary files. During the conversion, advanced developers can use a script called Quake C (*.qc) to describe specific parameters of the model, such as mass, inertia, bounding volume model, etc. If these parameters are not specified, the compiler chooses default values. The default bounding volume is a single convex shape that encompasses the geometry of the model.

4.5 Authoring virtual laboratories by Source SDK

The game engine has the advantage of providing a complete VE that usually not only includes mechanical devices but also non-mechanical elements such as a laboratory room, decorations and furniture, NPCs, etc. to enhance the immersive feelings for the players. Such VEs can be completed by a tool called ‘Hammer Map Editor’.

The ‘Hammer Map Editor’ is a part of the Source SDK. Through this editor, which has an interface similar to that of 3ds Max, virtual laboratory developers are able to build their own customized laboratory buildings and rooms in the same way as game developers design ‘game levels’. The ‘block tool’ provided by the editor is commonly used to create basic elements of rooms, such as walls and doors that cannot be moved inside of the VE. These elements are usually made of cubes. Another type of tool, called ‘entity tool’, can be used to place moveable decorations and furniture such as laptops and chairs into the virtual laboratory. These entities can be geometrically complex, and they can be modeled and imported into the VE through the process explained in the previous section.

The VE is saved in a Valve Map file (*.vmf). This type of file cannot be executed directly, and it must be compiled by the Hammer Map Editor. The compiled file includes a Valve Binary Space Partition file (*.bsp), a Valve Visibility Process file (*.vis) and a Valve Radiance file (*.vrad). These files can be loaded by the ‘Source’ game engine directly, and in combination they can facilitate the implementation of a virtual laboratory in GMod.[3]

4.6 Design and implementation of storylines

In order to offer successful educational laboratories, besides immersive VEs, player-controlled avatars and mechanical devices, well designed storylines are required. In the virtual assembly training system proposed here, a storyline is used to familiarize the trainees or students with the immersive VE, to help them to understand assembly processes, to convey background knowledge, to complete actual laboratory experiments and to observe the corresponding experimental results.

In a laboratory, the storyline is presented through two types of content: the challenge content and the instructional content. The challenge content includes tests and tasks. The tests are usually in the form of multiple choice tests and are used to assess the students’ theoretical and fundamental knowledge. In the current laboratory exercises, the students complete a pre-test, an after-experiment test and a post-test. The pre-test is used to measure the prior knowledge and level of preparation of the students. The after-experiment test includes experiment-specific questions for which the students can obtain the answers directly from the assembly itself or from the simulation of the assembly. The post-test was designed to let the students consolidate the knowledge acquired from the laboratory exercise. Tasks during experiments are designed with the goal of enhancing the students’ knowledge. Guided by the VE laboratory, the students must complete these tasks in a pre-set order. For instance, in the planet gear train exercise, the first task is to compute the radii of the gears from the respective module numbers and the numbers of teeth so that matching gear sets can be selected and assembled in the next step. The instructional content is comprised of tutorials that lead the students through the completion of the laboratory exercise. Following these instructions, the students can perform the experiment step by step.

[3] In fact, these files can be directly loaded by any ‘Source’ engine based games such as Counter-Strike and Half-life 2.

Pop-up windows are the main medium for giving instructions and assigning challenges. In GMod, pop-up windows are scripted by calling functions of the Valve Graphic User Interface (VGUI) library using the Lua script language. This library provides the fundamental elements of a graphic user interface, such as text, buttons, images, etc. Through these pop-up windows, storylines can be presented.

4.7 Chapter summary

In this chapter, an application of the framework for game-engine based assembly simulation was introduced. This application utilized a specific game development platform, GMod, to implement a virtual mechanical educational laboratory designed for assembly procedure simulation.

There are two major parts of this chapter. The first part of analyzed the pros and cons of the 'Source' game engine and concluded that it is suitable for implementing virtual laboratories. This part also analyzed the benefits of using GMod as implementation platform, which provides access to the Lua script language. Lua is an easy-to-learn language, which can be used to develop code for the laboratory system directly, thus avoiding the redundancies of coding based on the 'Source' game-engine, which relies on C++.

In the second part of this chapter, the implementation methods for a virtual laboratory system were discussed. The major tasks for implementing this laboratory system include map editing, mechanical part modeling and storyline development. These tasks were achieved with the help of various software packages such as Hammer Map Editor, 3ds Max, SolidWorks, etc.

Chapter 5 Educational VE-based laboratories for undergraduate education

5.1 Gear train laboratories authored via GMod

In the previous sections, the fundamental components of the virtual mechanical laboratory (namely models of mechanical devices, virtual laboratory environments and storylines) were explained. The methodologies for developing these elements of virtual laboratories were also presented. In the following sections, two cases of such a laboratory, one for simple gear train education and the other for planet gear train education, are described. The mechanical parts, assembly and assembly procedures, which are the core components of mechanical laboratories, are introduced. At last part of the chapter, student feedback and an analysis of the learning effectiveness are reviewed.

Developed based on the platform of GMod in conjunction with the assembly framework discussed in Chapter 3, these two laboratory exercises have already been administered to the students in the junior-level undergraduate course ‘ME358 Machine Dynamics and Mechanisms’ at Stevens Institute of Technology.

5.2 Mechanical components and assembly order of simple gear trains

The simple gear train laboratory is a fundamental exercise that was designed to help the students to understand the basics of gear systems by building and exploring a simple gear train. The gear parameters of module number, pitch diameter and number of teeth are addressed. Also, concepts such as gear ratio, idler gear, etc. are covered in the laboratory.

Among many parameters of gears, the module number and the number of teeth are the two most important parameters. Therefore, in the gear laboratory, instead of the radii of the gears, the module numbers and numbers of teeth of the gears are provided to the students, who then must convert these parameters into the radii by the equation:

$$r = MT \tag{7}$$

In the equation, *r* is the pitch radius, *M* is the module number and *T* is the number of teeth.

Figure 13 shows the components of a simple gear train. A variety of gears with different numbers of teeth and the same module number are given to the students to assemble different gear trains that have different gear ratios, input/output directions and center distances of input/output shafts.[4] The challenge to the students in this laboratory exercise is to select correct gears that satisfy the given design requirements. For instance, these requirements can be that the gear ratio must be equal to 0.4 and the output direction must be opposite to the input direction.

Between two gears, a gear engagement constraint, a kinematic constraint as described in Section 3.3.4, is applied conditionally. This constraint facilitates the simulation of paired gears by forcing the output gear to rotate in accordance with the angular velocity of the input gear and the given gear ratio. The angular velocity, gear radius, number of teeth and gear ratio of paired gears satisfy the following relationships:

$$R = \frac{\omega_A}{\omega_B} = \frac{r_B}{r_A} = \frac{N_B}{N_A} \tag{8}$$

In the equation, R is the gear ratio, ω_A, r_A and N_A are the angular velocity, pitch radius and number of teeth of the input gear and $\mathrm{\omega_B, r_B}$ and $\mathrm{N_B}$ are the angular velocity, pitch radius and number of teeth of the output gear, respectively.

As the gear engagement constraint is a kinematic constraint that does not regulate the relative assembly positions of two paired parts, it only takes effect when certain criteria are met. In this case, the center distance of two gears and the pitch radius of these gears must satisfy:

$$d = r_A + r_B \tag{9}$$

In the equation, r_A and r_B are the pitch radii of the input/output gears, respectively.

Of course, for a game-engine powered immersive laboratory, providing only gears is not enough because gears do not ‘float in the air’. Therefore, support components are provided and the students are required to assemble these components along with the gears. In this laboratory exercise, the

[4] At the current stage, the module numbers of all gears are set to be the same to simplify the laboratory exercise.

support components include a base, shaft holders and shafts. A shaft holder is connected to the base by a ‘slider constraint’ that allows the shaft holder to only translate along the T-slot on the base. A shaft is supported by a shaft holder at each of its ends, and a gear is mounted on a shaft.

The sequence of an assembly procedure is also enforced. In this laboratory exercise, the students are required to assemble the gear mechanism in a predetermined order that is the same as in a real situation. The predetermined assembly order is as follows:

1. Mount a shaft on a shaft holder.
2. Insert the shaft into the central hole of the gear.
3. Connect the other shaft holder on the shaft to form a gear-set sub-assembly that consists of two shaft holders, one gear and one shaft.
4. Mount the gear-set sub-assembly onto the base. The position of the gear-set is designated by students via a pop-up window.
5. Repeat these steps to assemble multiple gear-sets on the base to form a simple gear train with several gears.

In this assembly sequence, step 1 and step 2 are interchangeable. Also, the VE allows the students to make mistakes by omitting step 2, in which case the gear-set sub-assembly contains no gear, resulting in a dysfunctional gear train.

However, in a real situation, a gear must be mounted on a shaft before the shaft is connected with the holders on its two ends. That is, step 2 and step 3 are not interchangeable and the VE should automatically prevent the mounting of a gear on a shaft when both shaft holders have already been connected with the shaft. When assembling a gear with a shaft, the VE checks whether both ends of the shaft are occupied. If one or both ends are vacant, the assembly is allowed. Otherwise, the assembly request is rejected. The logic rule of the assembly sequence is:

$$A(ShaftGear) = \neg(O(ShaftLeft) \wedge O(ShaftRight)) \tag{10}$$

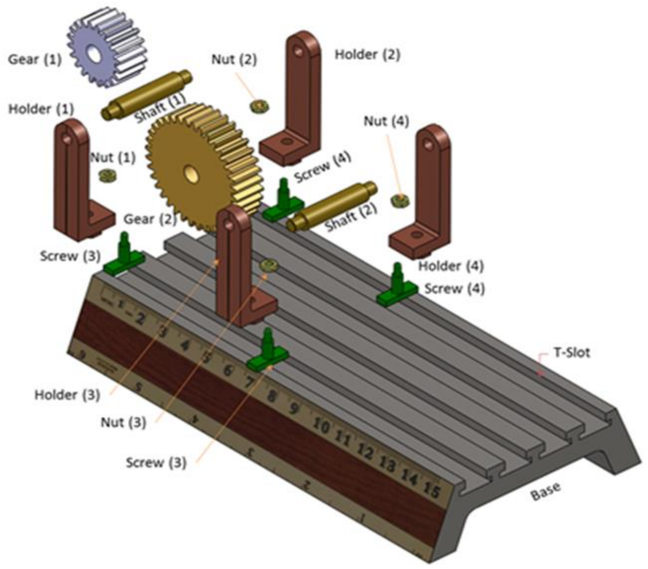


Figure 13: Major components of a simple gear train

When mounting a shaft holder onto the base, the exact position of the holder must be provided by the student. After this position has been assigned, an overlap check is performed to ensure that the holder and the gear connected to it do not overlap with other holders.

Since this laboratory exercise is offered simultaneously to multiple students who work as a team, there may be conflicts if multiple students assign the holder’s position at the same time. Therefore, an interlock algorithm that only allows one student at a time to assign a position was designed and implemented. When one student mounts a shaft on the base and tries to designate the position of the shaft holder, the other students are prevented from also mounting the shaft holder onto the base. Instead, they can only give advice to the student performing the assembly.

5.3 Mechanical components and assembly order of planet gear trains

A planetary gear train is shown in Figure 14. It is composed of a base, a ring gear, a planet carrier, four pins, four planet gears and a sun gear. Among these mechanical parts, the ring gear, the sun gear and the planet carrier can serve either as input or output. Furthermore, the dimensions of the sun gear, the ring gear, the planet gears and the planet carrier must satisfy the following constraints:

$$R_{ring} = R_{sun} + 2R_{planet}$$
$$R_{carrier} = R_{sun} + R_{planet} \quad (11)$$

In these constraint equations, the R's are the pitch radii of the gears or the radius of the carrier (i.e. distance from center of carrier to center of pin), respectively.

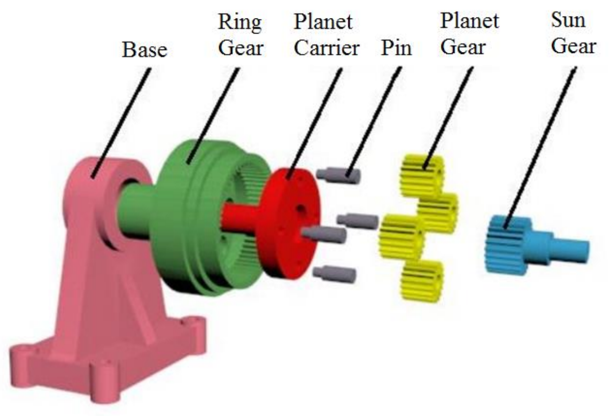


Figure 14: Components of a planetary gear train system

Besides the gears in the gear system, there are some support components that need to be assembled by the students. In this laboratory exercise, they are the base and the pins. The base is designed to hold all mechanical components and the pins, which have to be inserted into the planet carrier and serve as the shafts that hold the planet gears. Ideally, the students are also required to assemble all the components in a predefined order.[5] The correct order of the assembly is as follows:

1. Mount the planet gears on the pins.
2. Insert the pins into the planet carrier.

[5] In the practice of the student laboratory, the assembly order is not enforced as it would add to the complexity of the laboratory operations, which may result in the students not being able to complete the laboratory exercise in the limited time. However, some assembly sequence rules, for instance, step 3) of the list must be carried out before steps 4) and 5) as the operations in steps 4) and 5) are based on the successful assembly of the ring gear.

3. Connect the ring gear with the base by inserting the hollow cylinder portion of the ring gear into the cylindrical hole on the base.
4. Insert the cylindrical portion of the planet carrier, which is carrying the planet gears, into the cylindrical hole on the ring gear.
5. Mount the sun gear on the center of the planet gears to complete the assembly.

5.4 Storyline and laboratory activities

5.4.1 Simple gear train laboratory

There are four tasks in the simple gear train laboratory, as shown in Figure 15. The first task students must complete is assembling a gear set including two shaft holders, a shaft and a gear of their choice. The second task is to build a meshing gear pair. In order to complete this task, students must review the relationship of pitch radius against the module number and number of teeth, so that they could place gears at correct locations to let two gears mesh. The third task is to insert an idler gear between the previous two meshing gears to make a three-gear simple gear train. Then, a three-question multiple-choice test about this train is given by pop-up window. Students are allowed to answer collaboratively via messaging inside the virtual laboratory. The last task is to let students build a gear train with given requirements. Students must carefully review the requirements, discuss the gears they may use and build the train together.

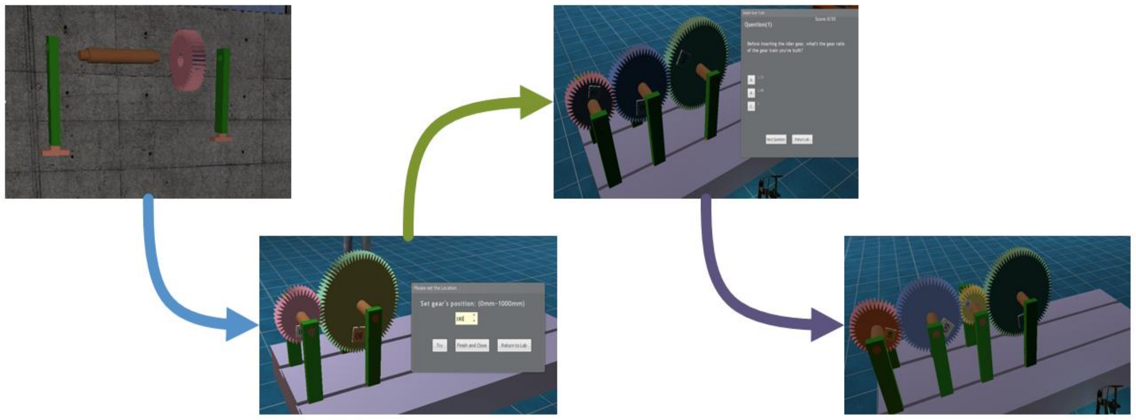


Figure 15: Process of simple gear train laboratory exercise

5.4.2 Planetary gear train laboratory

Like the simple gear train laboratory, the planetary gear train laboratory also includes a warm-up test. Then, students began a three-task experiment, the tasks of which are shown in Figure 16.

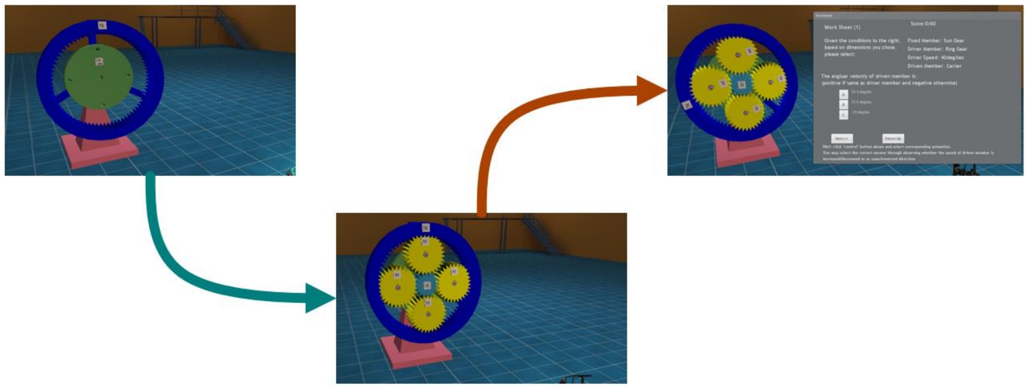


Figure 16: Tasks of planetary gear train laboratory exercise

At the first task, students were required to assemble the ring gear and the carrier. Then, they must put pins on the carrier and assemble the planet gears and a sun gear. Finally, they were required to answer 3 questions regarding the speed ratio of the gear train they assembled. They could answer the questions through observing the kinematics simulation of the gear trains.

5.5 Evaluations for usability and learning effectiveness

5.5.1 General description

In the spring semesters of 2013 and 2015, all students who took the course “Machine Dynamics and Mechanisms” at the authors’ institution were mandated to participate in the videogame-based laboratory exercises. These students were all mechanical engineering majors. Most of them were in their junior year, while a few of them were in their senior year.

Before tackling the textbook chapter on gears, the students had completed the study of the fundamentals of machine, four-bar mechanisms and cam design.

The students could either perform the experiment alone or collaborate with another student. All students used separate computers, no matter whether they worked alone or in a group. The students working in a group were not allowed to contact each other physically while performing the laboratory exercise.

The laboratories were evaluated with regards to their usability and learning effectiveness. The usability evaluation was conducted in 2013. It was targeted toward understanding whether students are able to adapt to the operations in this new form of laboratory. Because the laboratory is based on a popular game engine, it is also desirable to examine whether the students' previous exposure of relevant video games (or lack thereof) has an effect on their laboratory performance.

The learning effectiveness evaluation, which was conducted in 2015, was aimed at studying whether the laboratory exercises help students in understanding the required course materials. A short comparison of learning outcomes from the laboratory exercises and traditionally offered paper-based laboratory exercises was performed.

5.5.2 Timeline of evaluations

In both semesters, the laboratory exercises were scheduled by appointment outside of the class time in the school’s computer laboratory. The timeline of the lectures and laboratory exercises is listed in Table 4.

Table 4: Timeline of evaluation

Week	Activity	Note
7	Taking videogame background survey	Students take a short survey via surveymonkey.com outside of class
8-9	Attending gear train lectures	Students learn fundamentals of gears, simple gear trains, compound gear trains and planetary gear trains
10-11	Attending simple gear train laboratory	Students perform simple gear train laboratory exercise in groups
12-13	Attending planetary gear train laboratory	Students perform planetary gear train laboratory exercise in groups
14	Taking evaluation survey	Students take a survey on their satisfaction with virtual laboratories

5.6 Usability evaluation

5.6.1 Students’ gaming background survey

The evaluation of the laboratory exercises included a survey on the students’ videogame playing background, a pre-laboratory test, the experiment itself, a student satisfaction survey, and a final exam. The videogame background survey was composed of 3 questions aiming to evaluate the students’ gaming background including the frequency with which they played games, their exposure to various game genres and their familiarity with FPS games.

Form the survey, it was seen that only a few students had very minor prior videogame playing experience. In addition, there was an obvious difference between female and male students in playing games as most of the male students (74.36%) had played 20 minutes/week or more for some periods in their lives while most of the female students had played only a few games. The statistics are summarized in Table 5. 94 students participated in the usability study.

Table 5: Frequency of videogame playing of students

Gender	> 2 hours/week	>20 min/week; <2 hour/week	A few times only	Never or Almost never
Male	46.15%	28.21%	23.07%	2.56%
Female	6.25%	12.50%	50.00%	31.25%
Total	39.36%	25.53%	27.66%	7.45%

Since the basic operations of the virtual laboratory system such as moving, turning, communication, etc. are the same as those in most FPS games, a question concerning the students' FPS game playing experience was also posed. The results are listed in Table 6. It turned out that, among the 94 students, 70 students (74.47%) knew how to play FPS games. Since the laboratory exercises only require basic videogame playing skills, these students were classified as 'experienced' while the rest were considered 'inexperienced'.

Table 6: Students' previous exposure to FPS games

	Played very well	Knew basic operations	Tried few times only	Never played
	(experienced)		(inexperienced)	
Student	52.13%	22.34%	6.38%	19.14%

5.6.2 Assembly actions by students

In the laboratory exercises, the number of each student's assembly actions was recorded. Here, an 'assembly action' is defined as any operation in which a student successfully connected two parts into a sub-assembly. For example, connecting a shaft and a holder in the simple gear train laboratory exercise is an assembly action.

The first concern to be analyzed was whether the videogame playing background of the students affected their participation in the experimental procedures. Table 7 lists the students' number of assembly actions taken and their respective game playing background. Un-paired t-tests show that for both laboratory exercises, the students' previous exposure to first-person-shooting games did not have a significant impact on students' assembly performance (*p*>0.05).

Table 7: Students' number of assembly action vs their videogame background statistics

	Simple gear train laboratory		Planetary gear train laboratory	
	Average actions taken	Standard deviation	Average actions taken	Standard deviation
Experienced game player	22.21	12.13	14.25	15.57
Inexperienced game player	20.52	11.57	12.61	13.93
p-value (t-test unpaired two tail)	0.55		0.65	

The histograms comparing the distributions of number of assembly actions taken by experienced vs. inexperienced students are shown in Figure 17. The figure also indicates that those students who had little or no prior exposure to FPS games could perform similarly well compared to those who are experienced in these games. Therefore, it can be concluded that the laboratory exercises had good potential for learning as the inexperienced players were also able to acquire the basic skills for operating the virtual laboratory system through the tutorial without major struggles.

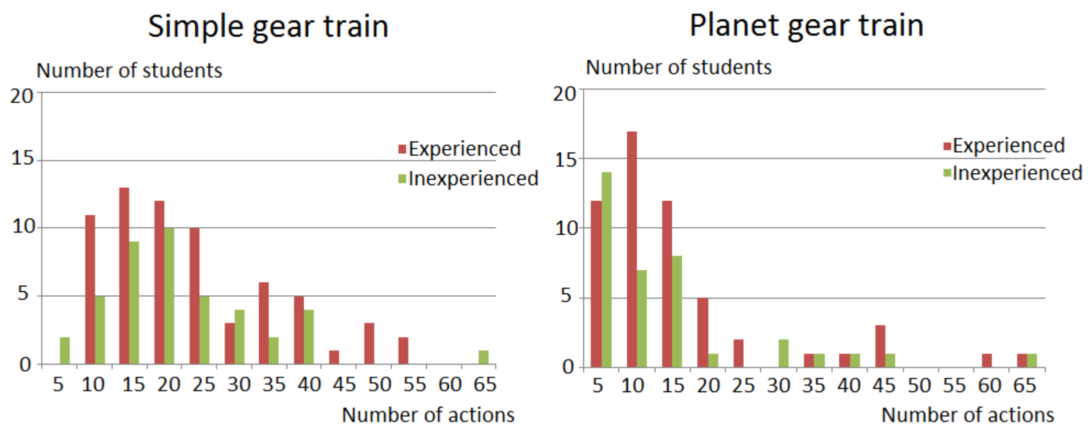


Figure 17: Comparison of number of actions vs. student background

5.6.3 Pre-laboratory test scores and final grades

In the simple gear train laboratory exercise, if no mistakes were made, each student group had to perform at least 16 assembly actions in order to complete all tasks. In the planetary gear train laboratory exercise, this number was 11. If the students made any mistakes such as choosing the wrong gears or assembling them in an incorrect sequence, their number of actions would be larger, sometimes even significantly larger.

Using the number of extra assembly actions, how the students' learning background may affect their laboratory exercise performances besides their prior game playing experience can be assessed. It is possible that those students who exhibit a better course performance may also perform better in the laboratory exercises. Therefore, the students' pre-laboratory tests scores and final grades were gathered to examine whether there was any correlation between the students' lecture course and laboratory performances. Here, the grades were not used to assess the learning effectiveness of the laboratory exercises.

However, as was the case regarding the students' prior game playing experience, there was no obvious correlation between the students' learning in the lecture portion of the course and their performance in the laboratory. The left part of Figure 18 depicts the students' number of extra actions vs. their final grade (in percent). The graph in the right part of the figure shows the number of extra actions each student performed vs. their pre-test grade (in percent). Since the students had completed their study of the textbook chapter on gears before the laboratory exercises, a large portion of them earned the maximum of points in the test.

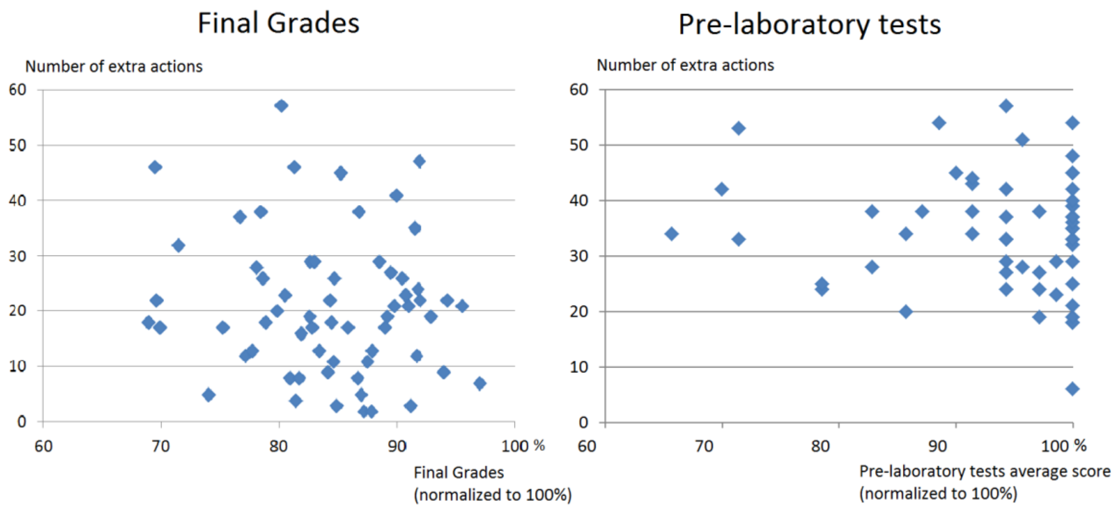


Figure 18: Comparison of number of actions vs. student learning

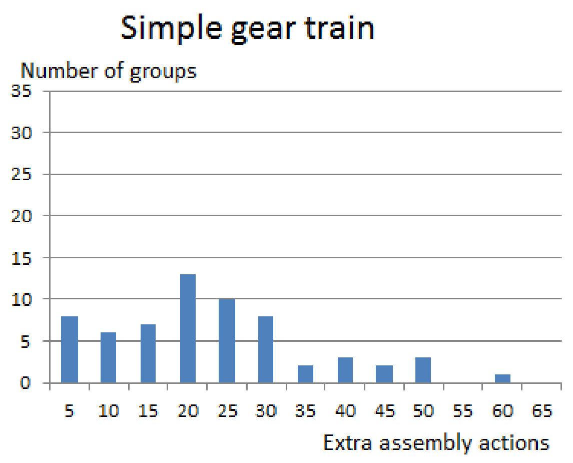
5.6.4 Students' group activities in virtual laboratory

By examining the number of extra assembly actions that each group had performed, the ease of use of the laboratory system could also be evaluated. Table 8 lists the overall average, standard deviation, maximum and minimum of the extra number of actions for all groups or individuals. From the table, it can be seen that on average each group took twice the minimum number of actions necessary, that is the students required several trial assemblies before completing all the tasks. It can also be noted that the average overall number of extra actions fell from the simple gear train to the planetary gear train exercises.

Table 8: Overall number of extra actions for laboratories

Laboratory	Average	Standard deviation	Maximum	Minimum
Simple gear train	19.84	12.74	57	2
Planetary gear train	11.93	18.11	74	0

As indicated in Figure 19, the distributions of the numbers of extra actions were quite different for the two laboratory exercises. For the simple gear train laboratory exercise, the distribution was relatively even for each horizontal-axis section. In the planetary gear train laboratory exercise, there were 26 groups or individuals (48.15%) that completed all tasks without any extra actions; while in the simple gear train laboratory exercise, there were none. From a conceptual knowledge perspective, planetary gear trains are much more difficult than simple gear trains. Therefore, it is likely that the reduction in the number of extra actions and the differences in their distributions were caused by the sequence of the two laboratory exercises. In the simple gear train laboratory exercise, the students may have been neither familiar with the operation of the laboratory system nor with the fundamental knowledge on gears. Therefore, a lot of trials were performed for the simple gear train assembly. The increase in the number of groups without any extra actions also indicates that the laboratory exercises led to self-tutoring by the students. That is, most of the students could remember the operation of the laboratory system and thus were able to avoid making similar mistakes as in the previous experiment.



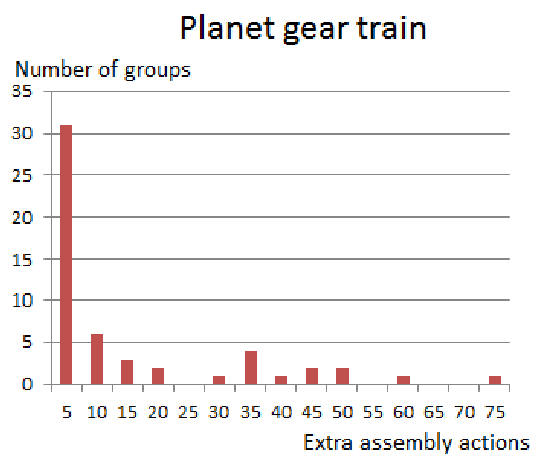


Figure 19: Histogram of number of groups by extra assembly actions

5.6.5 Conclusions of usability evaluation

From the evaluation results, it was observed that students were able to complete all experimental tasks for both laboratory exercises, regardless of their prior videogame playing experience. From the number of assembly actions performed by each student, it was also discovered that the laboratory exercises have good learning potential as the inexperienced videogame players were able to perform almost the same number of assembly actions as the experienced players in a limited time period. It was also found that the students' learning background did not exhibit a significant correlation with their laboratory performance. From comparisons of the students' performance between the two laboratory exercises, it could be seen that the laboratory exercises also lend themselves to memorization as most of the students made fewer mistakes in the second laboratory exercise despite it being more difficult in principle. Finally, the students' evaluation surveys indicate that most of the students were quite satisfied with the virtual laboratory exercises in general, their perceived learning effectiveness and the ease of operating the laboratory system.

5.7 Learning effectiveness evaluation

5.7.1 Measurements

The learning outcomes of the virtual laboratories were measured by the integration of:

1. the students' scores in pre-laboratory tests and post-laboratory tests
2. the students' final exam scores regarding the gear chapter in the textbook

The pre-and post-laboratory tests, consisting of 6 multiple choice questions each, were given immediately before and after each laboratory. These questions were designed to evaluate the students' understanding of very basic concepts of gear trains. In addition, in order to better evaluate the students' gain from conducting the laboratory exercises, the pre- and post-laboratory tests are designed based on a same set of course-covered knowledge topics, with different questions as well as altered order of the topics.

5.7.2 Overall pre- and post-laboratory test outcomes

The learning outcomes of these two laboratories were measured by the pre- and post-laboratory tests. As mentioned earlier, each test contains 6 multiple choice questions. Each question counts for 1 point. Figure 20 shows the average score and standard deviation of all tests.

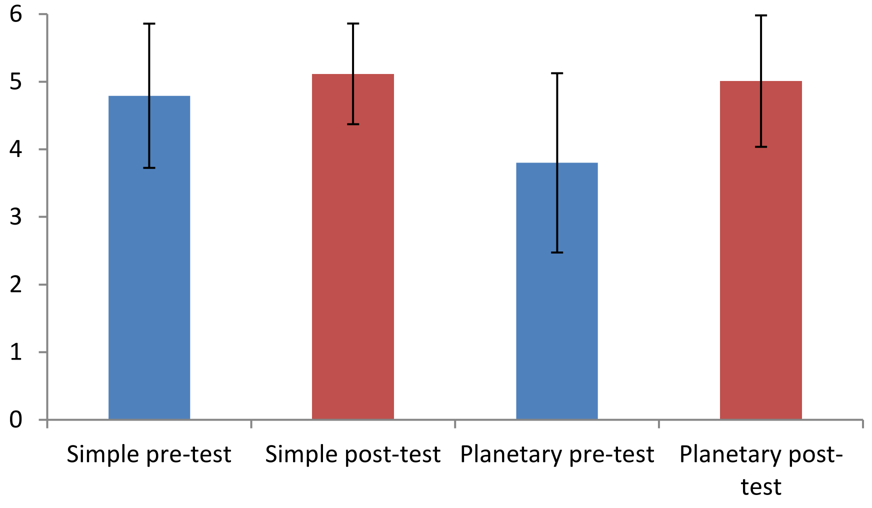


Figure 20: Means and standard deviations of scores for pre- and post- laboratory tests

From the tests of 107 students who performed the simple gear train laboratory exercise, the average scores of pre- and post-laboratory tests were 4.79 and 5.12 (a 6.9% increment), respectively. The standard deviations were 1.07 and 0.74 (a 30.8% decrement), respectively. The p-value from a paired two-tail t-test was 0.0035, showing that the improvement was statistically significant (p<0.05). There were 110 students who conducted the planetary gear train laboratory exercise. The average score increased from 3.80 in the pre-laboratory test to 5.01 in the post-laboratory test (a 31.8% increment). The standard deviation decreased from 1.33 to 0.97 (a 27.1% decrement). The p-value from a paired two-tail t-test was less than 0.0001, showing that the improvement was strongly statistically significant (p<0.05). Therefore, from the test statistics, it could be seen that the students' understanding of the required topics was enhanced from an overall perspective.

5.7.3 Correct percentage for each topic

Because the pre- and post- tests of each laboratory share the same topics with different questions, by evaluating the result from each topic, the learning effectiveness of the laboratory could be analyzed.

Table 9 lists all topics of the tests of the simple gear train laboratory exercise. From the table, it can be noted that in all topics, except in topic 2, more students gave more correct answers in the post-test than in the pre-test. However, it was also found that only 46.30% students could correctly answer the question on topic 2 in the post-test while 93.52% students could answer the question on this topic in the pre-test. There are two reasons for this drop. First of all, topic 2 is not directly addressed by the laboratory. The question is about the definition and components of the simple gear train, which were only covered in the lecture. Secondly, a pitfall choice in the post-test, stating “all axles of gears of a simple gear train must be in a straight line” misled many students, because in the laboratory, students were required to design a simple gear train with all axles on a straight line. Besides question 2, the only question that did not showed a significant improvement was question 6, for which most of the responses in the pre-test were correct already.

Table 10 lists the topics of the planetary gear train laboratory tests as well as the corresponding results. It can be seen that the correctness of the students’ answers increased in each of the topics. From paired t-test results, it can be seen that the improvements for 3 questions were significant while for the remaining 3 they were not.

A sign test was not conducted for each question because for each topic, a large number of students answered both the pre- and post-test questions correctly. The results of a one-tail sign test on the total scores for both the simple gear train and planetary gear train exercises are shown in Table 11. Half of those whose pre- and post-test scores were the same were counted as positive and the other half were counted as negative for sign-test computation purposes. The derived p-value showed that for both laboratory exercises, the improvement was significant (p<0.05).

Table 9: Test question topics for simple gear train laboratory and results

Topic Number	**Topic**	**Question # in pre-test**	**Question # in post-test**	**Pre-test correct %**	**Post-test correct %**	***p* (t-test, paired two tail)**
1	Definition of module number	1	3	82.41%	100.00%	<0.001
2	Definition and members of simple gear train	2	4	93.52%	46.30%	<0.001
3	Center distance between input/output gears	3	1	83.33%	100.00%	0.002
4	Functions for idler gears	4	2	67.59%	100.00%	<0.001
5	Computation of gear ratios for simple gear train	5	6	63.89%	93.52%	<0.001
6	Input/output direction change regarding to the number of idler gears	6	5	85.19%	94.44%	0.088

Table 10: Test question topics for planetary gear train laboratory and results

Topic Number	**Topic**	**Question # in pre-test**	**Question # in post-test**	**Pre-test correct %**	**Post-test correct %**	***p* (t-test, paired two tail)**
1	Members of a planet gear train	1	5	79.09%	83.64%	0.0576
2	Speed of output gear under different configurations	2	3	60.91%	92.73%	<0.0001
3	Dimensions of members of planet gear train	3	2	61.82%	66.36%	0.4167
4	Relationship of torque and speed change	4	1	36.36%	99.09%	<0.0001
5	Velocity direction of output gear under different configurations	5	4	66.36%	67.27%	0.4772
6	Input/output gears speed changes for different configurations	6	6	75.45%	91.82%	0.0003

Table 11: Sign test for simple and planetary gear train laboratory exercises

Laboratory exercise	*# of students whose total score increased*	*# of students whose total score did not change*	*# of students whose total score decreased*	*p-value (one-side sign test)*
Simple gear train	47	35	25	0.0165
Planetary gear train	80	20	10	<0.001

5.7.4 Effect of virtual laboratories on students' final exam grade on gear chapter

In the author's school, the course "Machine Dynamics and Mechanisms" is offered on a semester basis. The virtual laboratories that were evaluated here were offered in the spring semester of 2015. In the fall semester of 2015, these virtual laboratories were not offered. Instead, paper-based practice laboratories were offered. With the same course contents, a similar set of homework assignments, the same teaching staff, and similar examination questions, the effect of the virtual laboratories could be evaluated. However, because many uncontrollable factors existed, such as detailed course schedule, the students' background (although the students were from the same school, in the spring semester, most students were in their junior year and in the fall semester, most were in their senior year), etc., such comparison could only be considered preliminary.

In the final exams of both semesters, two 20-point problems, with one being about a simple gear train and the other being about planetary gear trains, were posed. Although the questions were different, the topics tested were the same. Figure 21 shows the mean and standard deviation of the students' final exam grades on the gear chapter. 35 students in the fall semester, who did not perform the virtual laboratory exercises, on average obtained a total score of 30.69 on the 2 questions, with a standard deviation of 9.48. 107 students from the spring semester, who conducted both laboratory exercises, on average obtained a total score of 32.92, with a standard deviation of 6.58. There was an increment in the average score and a decrement in the standard deviation. However, based on the un-paired t-test that resulted in a value of p = 0.1235, the increment should be considered not significant.

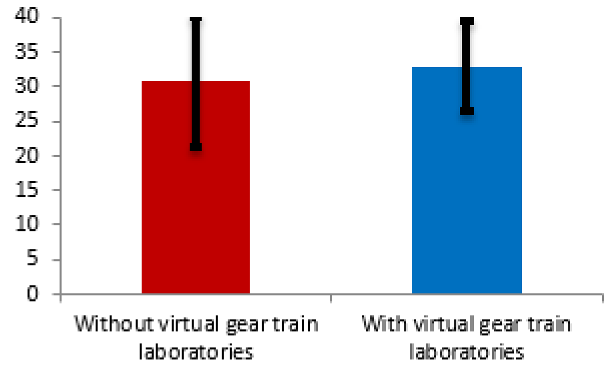


Figure 21: Comparison of mean and standard deviation for final exam grade on gear chapter

5.7.5 Students' self-reported collaboration and VE-recorded collaboration

It is also worth noting that surveys with 2 multiple choice questions about their collaboration were administered to the students immediately after each laboratory exercise.

1. What's your opinion of your group collaboration (5 – ideal, 3 – plain, 1 – nasty)?
2. How do you evaluate your workload (5 – I did most of the work, 3 – we evenly distributed the work, 1 – My partner did most of the work)?

The result is listed in Table 12. Most of the groups were formed voluntarily. 6 groups in the simple gear train laboratory and 4 groups in the planetary gear train laboratory were arranged by the teaching staff. It can be seen from the table that, regardless of how the group was formed, all students believed that they had an ideal partner. In addition, they all believed that they had evenly distributed their work, against the actual record acquired by the virtual laboratory system. This result may have been caused by the students' personal relationships. Apparently, the students did not want to report their true opinion at the risk of harming their personal relationships.

Table 12: Students' self-reported collaboration result for both laboratory exercises

Question number	1	2	3	4	5	# reported[6]
1	0.00%	0.00%	0.00%	0.00%	100.00%	96
2	0.00%	0.00%	100.00%	0.00%	0.00%	96

An advantage of the virtual laboratory system is its ability to track the students' participation. In traditional educational laboratories, an individual student's participation in a group project can only be evaluated through surveys, which are subjective. In the virtual laboratory, on the contrary, the evaluation could be accomplished by the computer automatically recording the students' activities.

In both the simple gear train and planet gear train laboratory exercises, each student's assembly actions were recorded. Here, an "assembly action" is defined as any operation in which a student successfully connected two parts into a sub-assembly.

For grouped work, a workload factor was defined as the ratio of the number of assembly actions by one member of a group to the number of assembly actions by the other group members, with the smaller number as the numerator. This workload factor can be used for evaluating the students' group collaboration. It would be 1 if two students balanced their work perfectly, and it would be 0 if one of the students did not contribute anything.

Figure 22 and Figure 23 depict the distributions of the workload ratio for the simple gear train and the planetary gear train laboratory exercises, respectively. In the figures, students who conducted laboratories individually were not counted. In the simple gear train laboratory exercise, the average workload ratio from 52 groups was 0.61, meaning that on average approximately one member performed 2/5 of all assembly activities while the other did 3/5. In the planetary gear train laboratory exercise, the average workload ratio was 0.51 for all 46 groups, dropping from the previous value of 0.61. The drop may have been caused by 2 reasons:

[6] Only groups with 2 members included. 2 groups did not report.

1. Some students gained more experience than their group members from the first laboratory; therefore, they could have completed the laboratory exercise with less collaborative effort.

2. The planetary gear train laboratory exercise required less effort than the simple gear train laboratory exercise; therefore, the students did not need to collaborate as in the previous laboratory exercise.

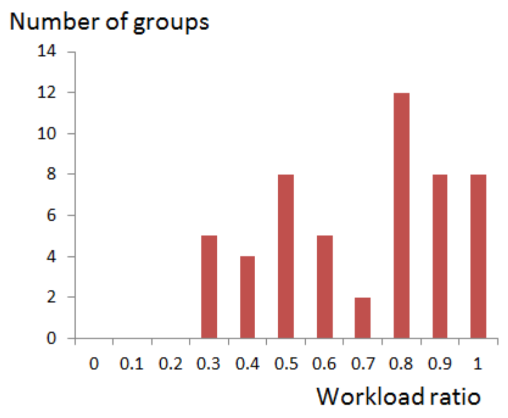


Figure 22: Workload ratio for simple gear train laboratory exercise

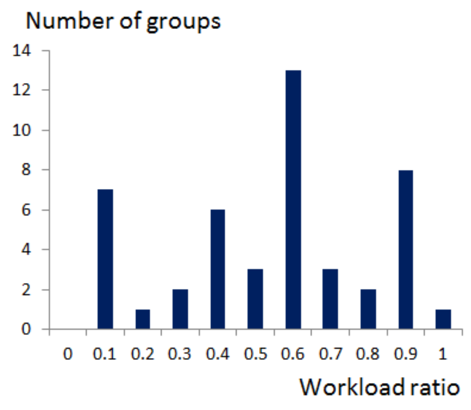


Figure 23: Workload ratio for planet gear train laboratory exercise

5.7.6 Conclusions on learning effectiveness and students' collaboration

Both laboratory exercises involve a pre-laboratory test and a post-laboratory test, which include the same topics with different questions, for evaluating the learning effectiveness. It was found that the students exhibited a better overall performance in the post-laboratory test than in the pre-laboratory test. The average scores in both post-laboratory tests were higher than those in the pre-laboratory tests. The standard deviation for the post-laboratory tests were both lower. From the comparisons of the grades of the questions on gears in the final exam between students who performed the virtual laboratory exercise in the spring semester of 2015 and students who did not conduct it in the fall semester of 2015, it could be observed that the students who performed the virtual laboratory exercise had higher average scores than those who did not.

From the students' collaboration data recorded by the virtual laboratory system, an overall imbalanced workload distribution among group members was found. In most of groups, one student contributed more than the other. However, from their self-reporting, the students all believed that they had balanced their workload well.

5.8 Student survey

In order to acquire student evaluations for the virtual laboratories, a paper-based anonymous survey was administered after the students had completed both laboratory exercises. The survey contained 3 multiple choice questions, which the students answered on a scale of 1-5, with 1 being the most negative and 5 being the most positive. The questions were:

1. What's your overall opinion on the virtual laboratories?
2. How do you evaluate the difficulty of operations?
3. Do you feel you learned something?

The result from 108 responses is listed in Table 13. From the table, it can be seen that on average the students gave a score of 4.42 on question 1, 3.78 on question 2, and 4.05 on question 3. From the survey, it can be seen that most students responded favorably to the overall concept of adapting games for creating educational laboratories. It can also be seen that the biggest concern in this form of laboratory is the operation. Many students faced challenges in learning how to operate the laboratory system. From a learning effectiveness perspective, the survey indicated that the students generally felt that the laboratory exercises helped them to better understand simple and planetary gear trains.

Table 13: Student survey results

Question	**1 – very negative**	**2 - negative**	**3 - neutral**	**4 - positive**	**5 – very positive**
1 Overall evaluation	0.00%	0.93%	16.67%	22.22%	60.19%
2 Usability	0.00%	5.56%	34.26%	37.04%	23.15%
3 Learning effectiveness	0.93%	3.70%	23.15%	34.26%	37.96%

5.9 Chapter summary

This chapter describes two educational VE-based laboratories that are authored based on the assembly representation proposed in chapter Chapter 3, and utilizes game engine as described in chapter Chapter 4. There are two parts for this chapter. In the first part of this chapter, two mechanical laboratory exercises, namely a simple gear train exercise and a planetary gear train exercise, were demonstrated. The mechanical devices were modeled and assembly rules such as gear engagement of the simple gear train and dimensional compatibility of the planet gear train were illustrated. The overall process of the laboratory exercises was introduced.

In the next part, two gear train laboratory exercises conducted on undergraduate mechanical engineering students was evaluated. The evaluation for both usability and learning effectiveness utilize data stem from three sources: test results, activity logs and survey questionnaires. From the data, it can be observed that the students were able to complete the assembly tasks in the game-based virtual laboratory, and the students were generally satisfied with the laboratory experience. From the comparison between the pre-laboratory and post-laboratory tests, it can be seen that there is a marginal improvement in the conceptual knowledge of the students. It can be found that, from the students' assembly activity logs as well as the surveys on the students', gaming background in order to better understand the usability and effectiveness of these laboratory exercises.

Chapter 6 Natural human-computer interaction

6.1 Why keyboard and mouse only?

Nowadays, personal computers are equipped with a keyboard and a mouse. They are so widely applied due to their versatility, effectiveness and low cost. One common application of personal computers is video games. A skilled video game player can fluently operate the computer using keyboard and mouse by simultaneously using both hands.

How can one become a skilled player then? Firstly, one must memorize the functionalities of each operation. In most first person shooting games, the basic operations, such as walking, firing, turning, etc., are the same. For instance, if one wants to slowly approach a street corner, hide there, then move around the corner and suddenly attack an enemy, one must (i) press the keys “w” and “alt” together in order to move forward slowly, (ii) release the “alt” key but not the “w” key in order to quickly move around the corner, and (iii) use the mouse to aim at the enemy and fire a shot as fast as possible. An experienced player may even press the ‘space’ key to jump around the corner rather than moving slowly around the corner in order to confuse the enemy, while still being able to simultaneously aim at the enemy by mouse. Without long-term practice, it is hard to manage all these skills.

For a videogame-based virtual laboratory, requiring students to practice like in game playing is not practical. Besides, for VEs, the input methods used in the system greatly affect the players’ feel of immersion. For instance, in a gear train laboratory with real gears, students assemble and disassemble by using hands and tools, rather than by moving a mouse and pressing keys. What makes it worse is that students may be frustrated not because they could not understand the laboratory materials but because of their difficulties in interacting with the VE. Therefore, the question arises whether or not there is any alternative to keyboard and mouse.

New motion sensing input devices make this possible. Such devices use sensors such as infrared sensors, accelerometers, vision sensors, etc. that detect partial or full body movements of people to replace keyboards and mouse as input devices. For instance, the Nintendo Wii uses accelerometers, which are integrated into a hand-held console for playing various video games. In a tennis game, this hand-held console functions like a tennis racquet. It detects the acceleration of the player's arm for simulating the strength as well as the direction of hitting a ball.

The Microsoft Kinect is another popular motion sensing input device that was originally designed for gaming applications. It comprises a camera, an infrared sensor and an array of microphones. Compared with the Wii, the Kinect is more flexible and versatile not only because its use does not involve the hands, but also it enables the player to give commands by gestures and speeches.

Since the actions of performing experiments and creating assemblies in the real world rely on the hands and tools instead of mouse and keyboard, the Microsoft Kinect provides an alternative as input device for virtual laboratory implementations. For instance, students can use gestures to mimic assembly actions. Since the Kinect has a limited sensing range, in order to enable students to navigate in a VE, speech commands are introduced.

In this chapter, the possibility of adapting the Kinect as a new type of input device for VEs that can replace keyboard and mouse will be explored. Section 6.2and section 6.3 will illustrate the detailed planned work of avateering and speech commanding. By using the Kinect's skeleton tracking capabilities, the motions of a student can be synchronized with an avatar within the VE. Therefore, the student can smoothly control the avatar to pick up mechanical parts or tools in order to create an assembly. By using speech recognition, the students are enabled to give commands for navigation or menu selection that cannot be easily accomplished by gestures.

6.2 Avateering in GMod based VEs

6.2.1 Human ragdolls in game engines

Like all first-person or third-person shooting games, GMod provides sophisticated human avatars, which can either serve as game player characters or as NPCs. In the games, certain gestures such as walking, aiming, jumping, etc. are enabled by the game authors. However, the physical models of the avatars actually allow them to be adapted to make gestures in a flexible manner.

The models of these avatars can be animated and are referred to as ‘ragdolls’ by the developer community. A ragdoll model has two types of meshes associated with it. The first type is the geometry mesh for rendering, and the other type is the physics mesh for physics simulation. Figure 24 shows a bone model (left) and the rendering mesh (the blue mesh) as well as a part of the physics mesh (the pivot, the red mesh) on the right. The term ‘ragdoll’ indicates that, like dolls, these models have virtual bones that cannot bend and joints that allow connected bones to rotate relative to each other within certain limits. An avatar’s gestures are actually simulated by controlling the movement of its joints. For example, when simulating head nodding, the game engine simply causes the joint between the ‘neck bone’ and the ‘spine bone’ to flex and extend repeatedly.

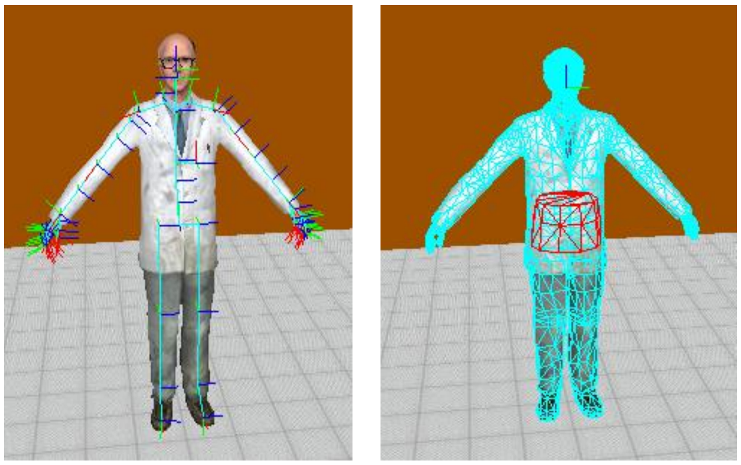


Figure 24: Bones and meshes of an avatar

However, synchronizing a human body and an avatar by controlling the ‘bones’ in GMod is unrealistic, because a human body ragdoll typically has around 70 bones. In addition, in GMod the bones are not directly accessible for game developers.

What can be accessed instead are the physics meshes. A ragdoll’s physics mesh is usually divided into several parts, each representing a portion of the avatar’s body and therefore containing several bones. Physics animations are based on the physics model rather than on the geometry model. Properties such as mass, moment of inertia, friction factor, etc., and collision models (i.e. bounding box) are assigned to each part of the physics mesh. By dividing a ragdoll’s physics mesh into multiple parts, these parts can then behave differently while they are still integrated in one entity, just like in a real person. For example, when an avatar is moving forward and its right hand hits a desk but its torso does not, the avatar can still move forward on the previous path with the hand changing its course, instead of the whole avatar stopping. Physics meshes are usually simpler than geometry meshes for simplification purposes as physics simulations and collision detection consume significant computation power.

6.2.2 Mapping gestures into GMod-based VEs

One of the most exciting features of the Kinect is its body tracking capability. Based on the depth data acquired, the machine learning algorithm that supports the Kinect is capable of extracting the positions of the human body’s major joints at very high rates and with promising accuracy. The Kinect SDK enables third party software developers to employ the Kinect for tracking the human body’s motion using the C# and C++ languages.

Currently, the Kinect SDK supports the tracking of a human body containing 20 joints: 4 for each arm, 4 for each leg, 2 for the torso, 1 for the head and 1 for the neck. The position of each joint in 3D skeleton coordinates *(x, y, z)*, which is a Cartesian coordinate system with its origin at the Kinect, can be determined in real time.

In GMod, there are 15 physics models for a human avatar ragdoll, and by setting their positions for each time frame, the avatar ragdoll can be animated. The physics mesh partitions on the human avatar ragdoll are correlated with the joints of the Kinect SDK skeleton, with major differences being that the former does not have ankle, wrist and neck joints. Furthermore, the positions of the ‘chest’ joint (the center position of one's chest and the pivot joint which is usually at the hip) are defined at different places. Figure 25 depicts a skeleton derived from the Kinect with red dots denoting the joints whose positions are used for avateering in GMod. Figure 26 shows an avatar ragdoll in GMod with red balls on the body denoting the avatar's joints whose positions are obtained from the Kinect while tracking the movement of a real person. It should be noted that these red balls are used only for illustration and debugging purposes, and during experiments, they are not rendered.

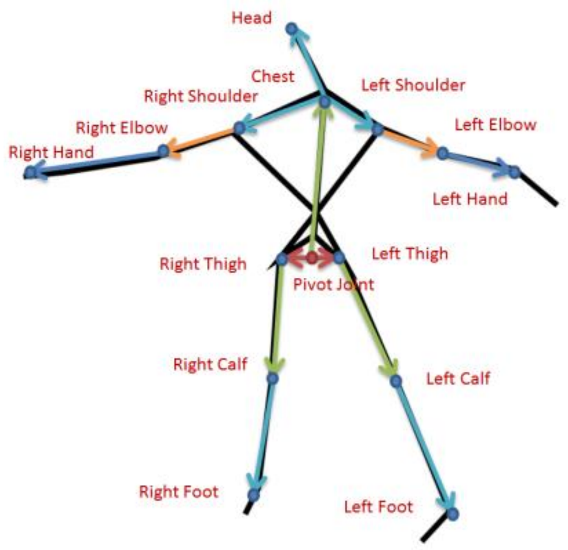


Figure 25: Skeleton derived from Microsoft Kinect

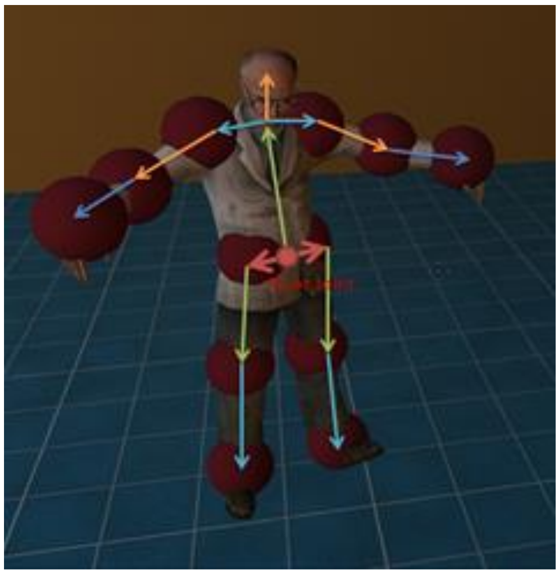


Figure 26: Avatar ragdoll in GMod

When the position of each physics model is assigned directly based on Kinect-acquired position information when animating the human avatar, the animation appears to be discontinuous. This is caused by the difference in the frame rate of the Kinect and that of the game engine. The internal rendering frame rate of the game engine is more than 200 fps; while that of the Kinect is only around 20 fps. What makes this situation worse is the fact that the communication between the Kinect SDK and the game engine also takes time. Thus, the skeleton frame rate that GMod can receive is further reduced. Therefore, to animate a human avatar, a better method is to specify the velocity of the selected physics model, as the game engine can automatically process the velocity and render the models at a higher rendering frame rate. The velocity for a physics model is:

$$v = \frac{P_t - P_c}{\Delta t} \tag{12}$$

In this equation, $\boldsymbol{P_t}$ is the target position that the Kinect SDK acquired from the actual person, $\boldsymbol{P_c}$ is the current position of the physics model and Δt is the time between two Kinect frames. Typically, Δt is 0.08 s.

Another challenge lies in the differences between the Kinect skeleton and the avatar. In the Kinect skeletal frame, the joint positions vary depending on the place where the human stands and so does the length of each bone. Also, the lengths of the bones extracted by the Kinect are disproportionate, i.e. the closer to the Kinect the bone is located, the longer it becomes.

The solution to this problem is converting the Kinect joint positions to bone orientations and normalizing the length to unity as follows:

$$N = \frac{P_{j1} - P_{j2}}{\|P_{j1} - P_{j2}\|} \tag{13}$$

In this equation, the ***Pj*s** are the positions of the two joints of a bone given by the Kinect SDK and the 3D vector ***N*** is the orientation of the bone. The position of a joint in the game engine is:

$$\tilde{P}_{j2} = \widetilde{P_{j1}} + L_b N \tag{14}$$

In this equation, $\tilde{P}_{j2}$ is the position of the joint to be derived, P_{j1} is the position of the known joint in the game engine and L_b is the length of the bone connected by these two joints. The lengths of these bones are pre-set by the designer who modeled the avatar.

The fundamental joint of the whole body is the pivot joint. The positions of all joints are represented relative to this joint as the origin. However, although the Kinect SDK provides the position of this joint, its position is derived by finding the center point of the left and right thighs. This is because in the game engine, the position of the pivot point is closer to the center of the thigh joints.

6.2.3 Translation and rotation of an avatar

In the assembly platform presented here, the avatar is capable of navigating around the virtual environment rather than only standing at a certain point. Therefore, the skeleton as a whole must be able to move in the plane and make turns. The best way to achieve this is to differentiate the world coordinate system and the avatar coordinate system.

The world coordinate system is fixed to the virtual environment. It is defined by the virtual environment map author and cannot be changed. The avatar coordinate system is attached to the avatar, with the *x*-axis pointing to the right, the *y*-axis pointing up and the *z*-axis pointing front. The origin point is on the pivot point. Since the turning always occurs about the vertical *y*-axis, a joint's position in the world coordinate system is obtained from its avatar coordinates as follows:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} \tag{15}$$

In this equation, a joint's position in the avatar coordinate is described by the (upper case) coordinates (X_1, Y_1, Z_1) where (x_p, y_p, z_p) are the pivot point's (small case) world coordinates and θ is the direction that the avatar looks at in world coordinate system.

The last question for the synchronization is how to enable multiple students to work collaboratively in the assembly environment. Since the platform that this paper presents is for multiple user online education, from network framework perspective, there are one server and several client computers. For a virtual assembly activity in which multiple students and instructors are participating, the position data are collected at the client computers, processed by both client and server computer, and finally sent back to the client side for rendering. The data flow diagram for the process described in this section is shown in Figure 27. The functions listed in the left (yellow) columns are executed on the client side and those in the right (blue) column are executed the on server side.

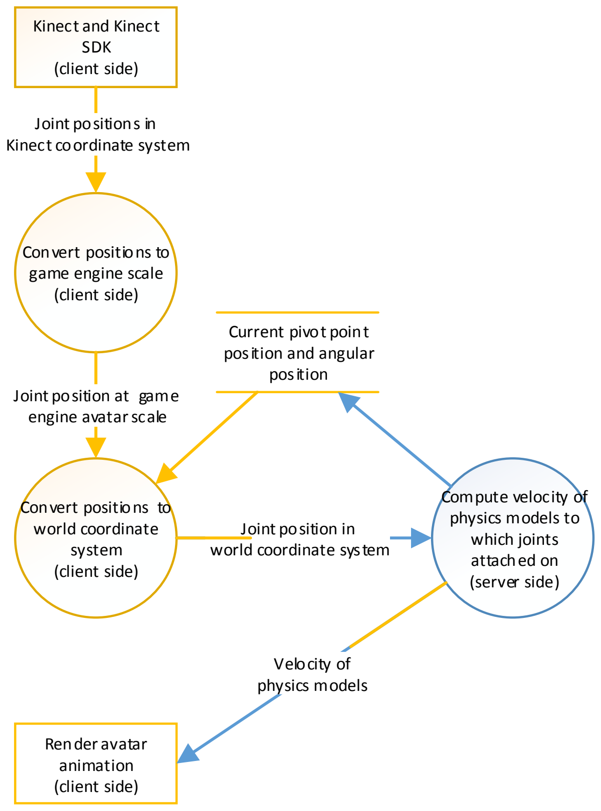

Kinect and Kinect SDK (client side)
Joint positions in Kinect coordinate system
Convert positions to game engine scale (client side)
Joint position at game engine avatar scale
Current pivot point position and angular position
Convert positions to world coordinate system (client side)
Joint position in world coordinate system
Compute velocity of physics models to which joints attached on (server side)
Velocity of physics models
Render avatar animation (client side)

Figure 27: Data flow diagram for motion synchronization

6.3 Speech commanding

6.3.1 Speech recognition and commanding

As mentioned in the previous section, in the virtual assembly system presented here, the students should be able to navigate through the 3D VE and complete tasks such as picking a part, answering questions, etc. without using a keyboard or mouse. Therefore, they need some method for commanding the avatars to move. The game engine supports a keyboard and mouse as the main input devices. Using these devices in combination with the Kinect is impractical since the students are not able to carry a keyboard or a mouse when they create an assembly by using their body motion.

Providing input commands using simple gestures or hand signals is a possible way to replace these traditional input devices [98] [99]. For instance, by traffic law, lifting the left arm to the left indicates the intent to turn left. This input method can be utilized in the VE, too. While for an application with only a few commands, this approach is acceptable, for an assembly platform that involves more than 10 commands, it is unlikely that the students can manage these gestures easily. Thus, the question is why commands must be given silently.

Using speech commands has the potential for greatly reducing the students' difficulties with getting started with the virtual assembly system. The Kinect contains an array of 4 microphones for collecting voice data. These voice data can then be processed and recognized by the Kinect SDK and by the Microsoft Speech API. The speech recognition can be accomplished at the same time as the body tracking.

A speech command is an order that students or instructors give. For instance, saying 'forward' causes an avatar in the VE to move forward. Generally, there are several types of commands such as navigation, GUI operation, assembly and choice menu operation. Examples of these commands are listed in Table 14.

Table 14: Examples of speech commands

Category	*Command*	*Speech*	*Command effect*
Navigation	Forward	Forward Forwards	Avatar moves forward
Navigation	Accelerate	Accelerate Faster	Avatar moves faster
Assembly	Drop	Drop Dropping	Drop a part from hands
GUI	Spawn	Spawn Create	Pop up list of parts
Choice selection	'A'	Alpha Apple	Select option 'A'

As can be seen in Table 14, a voice command consists of a command effect and several speeches. A speech is a word or phrase that the computer program can recognize from pronunciation. That is to say, the students or instructors speak these words or phrases to give a command. In order to render the commanding convenient, a command usually includes several speeches whose semantics are close to the meaning of the command in natural language. For instance, the words ‘accelerate’ and ‘faster’ both mean move faster. Thus, these two words were chosen here for the command of ‘accelerate’.

6.3.2 Voice command processing

Like motion tracking, the voice command processing involves both the server and client computers. A voice signal is detected and recognized on the client side. The client then reports the command to the server. The server responds to the command accordingly and sends its response back to the client for rendering. Since there are 4 categories of voice commands, the flow chart of processing these commands is more complicated than that for processing the body tracking information. A simplified flow chart is shown in

Figure 28. When a voice command is received, the assembly platform first checks whether there is currently a pop-up window on the student’s screen. If so, the server only accepts pop-up menu operation commands.

When a navigation command is received, the avatar moves on the floor or makes turns accordingly. When the avatar collides with an object that it cannot pass through, it stops. If the student still gives this navigation command, the avatar rejects it. For example, when the avatar moves forward and hits an obstacle in the VE, an additional “forward” command is not executed until the student orders “backward” to move the avatar away from the wall (see Figure 29 and Figure 30)

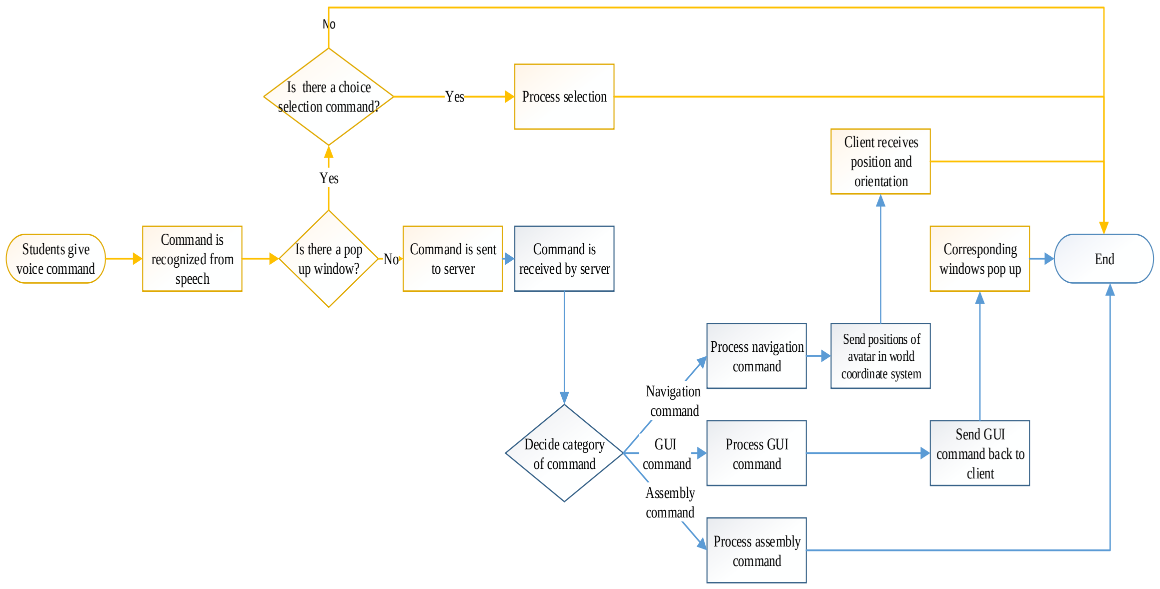

No
Is there a choice selection command?
Yes
Process selection
Client receives position and orientation
Yes
Students give voice command
Command is recognized from speech
Is there a pop up window?
No
Command is sent to server
Command is received by server
Corresponding windows pop up
End
Process navigation command
Send positions of avatar in world coordinate system
Navigation command
Decide category of command
GUI command
Process GUI command
Send GUI command back to client
Assembly command
Process assembly command

Figure 28: Flow chart for voice command processing

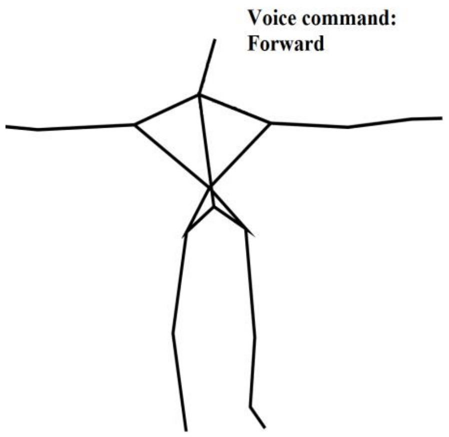


Figure 29: “Forward” command is received by Kinect

Figure 30: “Forward” command is not executed by the avatar

6.4 Part operation by gesture and speech

In the current system, a part can be picked and manipulated by one hand or by both hands. In this way, the picking appears to be realistic. Generally, for larger parts, two-hand picking is used while for smaller parts, one hand suffices.

The picking positions are virtual handles for avatars to hold during the assembly. Of course, these “handles” do not necessary have the shape of a handle. These picking positions are pre-defined by the assembly educator in the part’s local coordinates. Figure 31 shows a gear with two “handles” on its shaft for two-hand picking. A local coordinate system is attached to the right end of the shaft.

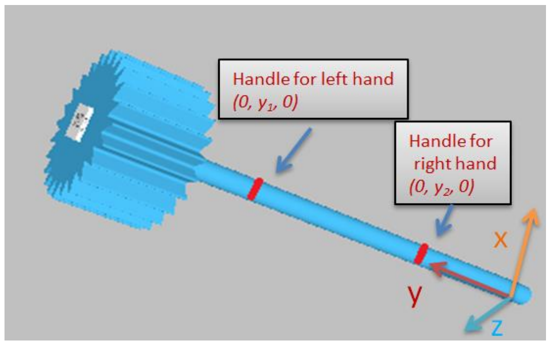


Figure 31: Part defined with two “handles” for two-hand picking

The picking procedure was simplified in the system described here. When the physical model of an avatar's hand hits a part, a picking-validation process is triggered. This validation process is show in Figure 32.

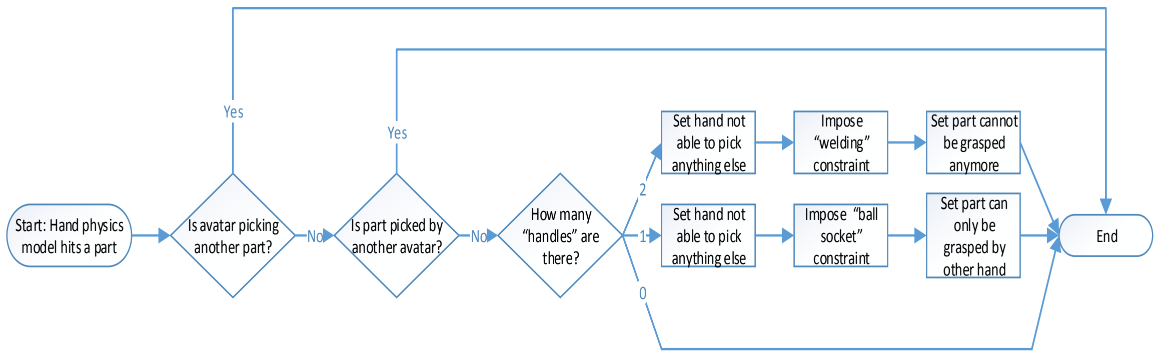

Start: Hand physics model hits a part
Is avatar picking another part?
Yes
No
Is part picked by another avatar?
Yes
No
How many "handles" are there?
2
1
0
Set hand not able to pick anything else
Impose "welding" constraint
Set part cannot be grasped anymore
Set hand not able to pick anything else
Impose "ball socket" constraint
Set part can only be grasped by other hand
End

Figure 32: Flow chart for validation process of picking

If the validation is passed, a kinematic constraint is imposed between the physics model and the part at the place of the "handle". For one-hand picking, the constraint imposed is a "welding" constraint, which removes all six relative degrees of freedom between the hand and the part. For two-hand picking, the constraint imposed is of the type "ball socket", which removes only the three relative translational degrees of freedom. In this way, the avatar can rotate the part with flexibility by grasping the part with both hands. Figure 33 and Figure 34 show an avatar using both hands to pick up a gear.

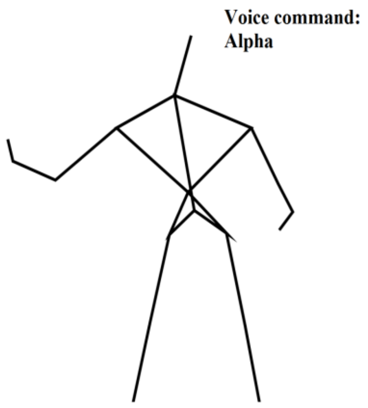


Figure 33: Skeleton tracked by Kinect for picking

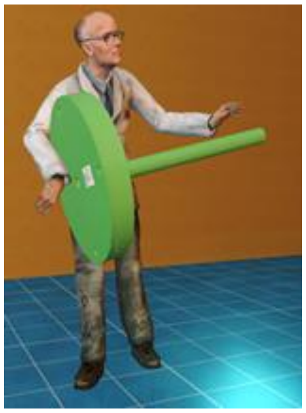


Figure 34: Avatar in GMod picks by both hands

6.5 Generating an assembly by Microsoft Kinect sensor

In order to simplify the assembly system, an assembly validation process is triggered whenever a part held by a student approaches or touches another part. Based on this validation process, the system decides whether these two parts are available to be mated. If these two parts have mating features that allow them to be connected, an auto-guidance process is triggered so that pre-defined constraints are imposed. At the same time, the avatar releases its hand(s), thus ceasing to grasp the part.

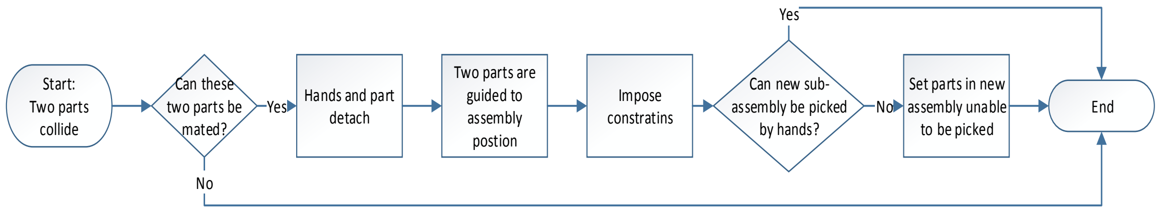

Start: Two parts collide
Can these two parts be mated?
Yes
No
Hands and part detach
Two parts are guided to assembly postion
Impose constratins
Can new sub-assembly be picked by hands?
Yes
No
Set parts in new assembly unable to be picked
End

Figure 35: Flow chart for assembly procedure

After the assembly has been completed, the assembly platform determines whether the new sub-assembly is eligible to be picked by this avatar. For instance, when a part of the sub-assembly is fixed to the ground, the new sub-assembly cannot be picked again. In some other cases, the sub-assembly can be picked again and manipulated. The flow chart of the assembly procedure is shown in Figure 35.

6.6 Chapter summary

In this chapter, a platform for mechanical assembly education was described. This platform consists of two main features: (i) a game engine as the foundation of desktop VR and (ii) the Kinect as the only input device. A prototype of this platform has been implemented through integration of C# based Kinect SDK and physics sand box GMod.

Under this platform, students who learn about mechanical assemblies can use their bodies' motion to control fully animated avatars and complete assembly tasks. They can also use voice commands to interact with computers. Compared to simulations controlled by keyboard and mouse, simulations with the Kinect interface are more realistic. In addition, compared to assembly simulations based on immersive VR technology, the platform described here is more affordable.

The technical details of the platform were described, the major challenges and solutions were presented and flow charts for these solutions were provided. These challenges include human-avatar synchronization, voice commanding, part picking and part assembly. Through the integration of these components, the platform exhibits robustness in creating various assemblies and ease in designing pedagogically effective assembly processes.

Chapter 7 Conclusions

This document introduces a framework for authoring VEs for mechanical assembly training. This framework is based on adapting a computer game engine originally designed for implementing entertainment environments for the development of educational laboratory environments. In order to make the game engine compatible with state-of-the-art engineering capabilities, a feature-based assembly representation was created. In accordance with the feature-based design concept, this representation includes assembly features, feature associations, kinematic constraints and sub-assembly hierarchies. The operations involved in an assembly process are discussed and an algorithm for judging the validity of such operations was described.

The main contribution of this work is that the framework presented here reduces the complexity and costs associated with developing virtual assembly training environments by using 3D game engine technology. This approach enables less experienced developers to design and implement similar applications. Even though game engines in general are not designed to support high levels of detail or dimensional accuracy, this framework expands the game engines' capabilities, thus allowing the implementation of mechanical assembly functionality. Furthermore, the approach described here enables developers of VEs to create game modifications that involve kinematic constraints between virtual objects without the need to access the game engine's source code. Therefore, this framework has the potential for being applied to industrial-level training and education.

As an example, a virtual laboratory for the assembly of gear trains was developed based on this framework. In the virtual laboratory exercises, student groups are tasked with assembling simple and planetary gear trains. They have to choose correct combinations of gear types and dimensions. Studies for this laboratory showed promising results in a usability evaluation and marginal learning improvements for students in a learning effectiveness evaluation. Although this was the first exposure of the students to such a virtual laboratory environment, they were able to complete the laboratory assignment. The level of collaboration recorded by the VE varied from group to group, while the self-reported levels of collaboration showed that all students agreed that they had collaborated perfectly.

In order to enhance the feel of immersion of the VE users, the Microsoft Kinect sensor was utilized. The Kinect sensor is able to capture one’s skeletal movement as well as speech. Therefore, this new sensor, combined with an existing game-based VE, can facilitate the students’ assembly by using their gestures rather than keyboard and mouse. A framework for using the sensor has been laid out.

Appendices

Appendix A Survey questionnaire for learning effectiveness evaluation

Name: DO NOT write your name on this survey

1. What's your overall reaction to the Game-based gear train labs? Please answer in the scale of 1-5.
    - ○ 5 (I like it, it's cool!)
    - ○ 4
    - ○ 3
    - ○ 2
    - ○ 1 (Completely waste of time)
2. How do you judge the difficulty of operations of the labs? Please answer in the scale of 1 – 5.
    - ○ 5 (It's easy to get started)
    - ○ 4
    - ○ 3
    - ○ 2
    - ○ 1 (It's really hard, I basically do not know what's going on)
3. Do you feel you learned something during the labs? Please answer on a scale of 1 -5.
    - ○ 5 (I feel my comprehension of gear and gear trains is enhanced)
    - ○ 4
    - ○ 3
    - ○ 2
    - ○ 1 (I learned nothing from it)

Bibliography

[1] R. Schroeder, "Defining virtual worlds and virtual environments," *Journal For Virtual Worlds Research*, vol. 1, no. 1, pp. 2-3, 2008.

[2] G. Boothroyd and P. Dewhurst, Product design for assembly, Boothroyd Dewhurst Incorporated, 1987.

[3] A. Seth, J. . M. Vance and J. H. Oliver, "Virtual reality for assembly methods prototyping: a review," *Virtual Reality*, vol. 15, no. 1, pp. 5-20, 2011.

[4] J. J. Shah and M. Mäntylä, "Parametric and feature-based CAD/CAM: concepts, techniques, and applications," New York, Wiley, 1995, pp. 53-78.

[5] C. . L. Jackins and S. . L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Computer Graphics and Image Processing,* vol. 14, no. 3, pp. 249-270, 1980.

[6] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129-147, 1982.

[7] H. B. Voelcker and A. A. G. Requicha, "Geometric modeling of mechanical parts and processes," *Computer*, vol. 10, no. 12, pp. 48-57, 1977.

[8] OpenGL, "Rendering pipeline overview," OepnGL.org, [Online]. Available: https://www.opengl.org/wiki/Rendering_Pipeline_Overview. [Accessed 13 06 2016].

[9] J. J. Shah and M. . T. Rogers, "Functional requirements and conceptual design of the feature-based modelling system," *Computer-Aided Engineering,* vol. 5, no. 1, pp. 9-15, 1988.

[10] T. Phong and S. Grewal, "A data model for an assembly planning software system," *Computer Integrated Manufacturing Systems,* vol. 10, no. 4, pp. 267-275, 1997.

[11] L. E. J. Brouwer, "On the foundations of mathematics," *Collected Works,* vol. 1, pp. 11-101, 1907.

[12] B. Smith, "Boundaries: an essay in mereotopology," in *The Philosophy of Roderick Chisholm*, 1997, pp. 534-561.

[13] B. Smith, "Mereotopology: a theory of parts and boundaries," *Data & Knowledge Engineering,* vol. 20, no. 3, pp. 287-303, 1996.

[14] K.-Y. Kim, "Ontology and assembly joint topology representation," *Compute-Aided Design and Applications,* vol. 5, no. 5, pp. 630-638, 2008.

[15] K.-Y. Kim, H. Yang and D.-W. Kim, "Mereotopological assembly joint information representation for collaborative product design," *Robotics and Computer-Integrated Manufacturing,* vol. 24, no. 6, pp. 744-754, 2008.

[16] W. F. Bronsvoort and A. Noort, "Multiple-view feature modelling for integral product development," *Computer-Aided Design,* vol. 36, no. 10, pp. 929-946, 2004.

[17] Y.-L. Lai, "A constraint-based system for product design and manufacturing," *Robotics and Computer-Integrated Manufacturing,* vol. 25, no. 1, pp. 246-258, 2009.

[18] M. L. Martínez and J. Félez, "A constraint solver to define correctly dimensioned and overdimensioned parts," *Computer-Aided Design*, vol. 37, no. 13, pp. 1353-1369, 2005.

[19] C. J. Barnes, G. F. Dalgleish, G. E. M. Jared, K. G. Swift and S. J. Tate, "Assembly sequence structures in design for assembly," in *Proceedings of the IEEE International Symposium on Assembly and Task Planning (ISATP97),* Marina del Rey, CA, 1997.

[20] R. B. Gottipolu and K. Ghosh, "A simplified and efficient representation for evaluation and selection of assembly sequences," *Computers in Industry,* vol. 50, no. 3, pp. 251-264, 2003.

[21] A. Banerjee and P. Banerjee, "A behavioral scene graph for rule enforcement in interactive virtual assembly sequence planning," *Computers in Industry,* vol. 42, no. 2, pp. 147-157, 2000.

[22] A. Swaminathan and K. S. Barber, "An experience-based assembly sequence planner for mechanical assemblies," *IEEE Transactions on Robotics and Automation,* vol. 12, no. 2, pp. 252-267, 1996.

[23] A. C. Lin and T. C. Chang, "An integrated approach to automated assembly planning for three-dimensional mechanical products," *The International Journal of Production Research,* vol. 31, no. 5, pp. 1201-1227, 1993.

[24] L. S. Homem de Mello and A. C. Sanderson, "Representations of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation,* vol. 7, no. 2, pp. 211-227, 1991.

[25] L. S. Homem de Mello and A. C. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation,* vol. 7, no. 2, pp. 228-240, 1991.

[26] J.-C. Yu and Y.-M. Li, "Structure representation for concurrent analysis of product assembly and disassembly," *Expert Systems with Applications,* vol. 31, no. 4, pp. 705-714, 2006.

[27] S. Zhang, W. Shen and H. Ghenniwa, "A review of Internet-based product information sharing and visualization," *Computers in Industry,* vol. 54, no. 1, pp. 1-15, 2004.

[28] Y. Oh, S.-h. Han and H. Suh, "Mapping product structures between CAD and PDM systems using UML," *Computer-Aided Design,* vol. 33, no. 7, pp. 521-529, 2001.

[29] L. Chen, Z. Song and B. Liavas, "Exploration of a multi-user collaborative assembly environment on the internet: a case study," in *Proceedings of the ASME Design Technical Conferences and Computers and Information in Engineering Conference(DETC'01)*, Pittsburgh, PA, 2001.

[30] L. Chen, Z. Song and L. Feng, "Internet-enabled real-time collaborative assembly modeling via an e-assembly system," *Computer-Aided Design,* vol. 36, no. 9, pp. 835-847, 2004.

[31] S. R. Ellis, "What are virtual environments," *Computer Graphics and Applications,* vol. 14, no. 1, pp. 17-22, 1994.

[32] W. R. Sherman and A. B. Craig, "Chapter 1- Introduction to virtual reality," in *Understanding Virtual Reality: Interface, Application, and Design*, San Francisco, Elsevier, 2002, pp. 1-16.

[33] Oculus, "State of the Art," Oculus, [Online]. Available: https://www.oculus.com/gear-vr/. [Accessed 13 06 2016].

[34] W. Zhao and V. Madhaven, "Virtual assembly operations with grasp and verbal interaction," in *Proceedings of the ACM International Conference on Virtual reality Continuum and Its Applications*, Hong Kong, China, 2006.

[35] B. M. Howard and J. M. Vance, "Desktop haptic virtual assembly using physically based modelling," *Virtual Reality,* vol. 11, no. 4, pp. 207-215, 2007.

[36] K. Walczak, W. Cellary and M. White, "Virtual museum exbibitions," *Computer,* vol. 39, no. 3, pp. 93-95, 2009.

[37] W. D. McCarty, S. Sheasby, P. Amburn, M. . R. Stytz and C. Switzer, "A virtual cockpit for a distributed interactive simulation," *IEEE Transactions on Computer Graphics and Applications,* vol. 14, no. 1, pp. 49-54, 1994.

[38] H. Gu, D. Wu and H. Liu, "Development of a Novel Low-Cost Flight," *World Academy of Science, Engineering and Technology,* vol. 60, pp. 685-689, 2009.

[39] D. Ota, B. Loftin, T. Saito, R. Lea and J. Keller, "Virtual reality in surgical education," *Computers in Biology and Medicine,* vol. 25, no. 2, pp. 127-137, 1995.

[40] P. J. Gorman, A. . H. Meier and K. . M. Thomas, "Simulation and virtual reality in surgical education: real or unreal," *Archives of Surgery,* vol. 134, no. 11, pp. 1203-1208, 1999.

[41] G. Riva, "Virtual reality in psychotherapy: review," *Cyberpsychology & Behavior,* vol. 8, no. 3, pp. 220-230, 2005.

[42] M. Krijn, P. M. G. Emmelkamp, R. . P. Olafsson and R. Biemond, "Virtual reality exposure therapy of anxiety disorders: A review," *Clinical Psychology Review,* vol. 24, no. 3, pp. 259-281, 2004.

[43] S. Jayaram, J. M. Vance, R. Gadt, U. Jayaram and H. Srinivasan, "Assessment of VR technology and its applications to engineering problems," *Journal of Computing and Information Science in Engineering, Transactions of the ASME,* vol. 1, pp. 72-83, 2001.

[44] M. J. Kim, X. Wang, P. E. Love, H. Li and S.-C. Kang, "Virtual reality for the built environment: a critical review of recent advances," *Journa of Information Technology in Construction,* vol. 18, pp. 279-305, 2013.

[45] J. Ye, R. I. Campbell, T. Page and K. S. Badni, "An investigation into the implementation of virtual reality technologies in support of conceptual design," *Design Studies,* vol. 27, no. 1, pp. 77-97, 2006.

[46] D. Weidlich, L. Cser, T. Polzin, D. Cristiano and H. Zickner, "Virtual reality approaches for immersive design," *CIRP Annals-Manufacturing Technology,* vol. 56, no. 1, pp. 139-142, 2007.

[47] E. Lin, I. Minis, D. . S. Nau and W. C. Regli, "Contribution to virtual manufacturing background research," Institute for Systems Research, University of Maryland, College Park, MD, 1995.

[48] T. S. Mujber,, T. Szecsi and M. S. J. Hashmi, "Virtual reality applications in manufacturing process simulation," *Journal of Materials Processing Technology,* vol. 155, pp. 1834-1838, 2004.

[49] DELMIA, "Assembly Process Simulation," DELMIA, [Online]. Available: http://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/DM-12857-Assembly-Process-Simulation-Datasheet-HR.pdf. [Accessed 13 06 2016].

[50] B. Y. Maiteh, M. C. Leu and D. L. Blackmore, "Virtual reality system for creation of design models and generation of numerically controlled machining trajectories," U.S. Patent Application 09/770,929, 2001.

[51] C.-S. Jun, K. Cha and Y.-S. Lee, "Optimizing tool orientations for 5-axis machining by configuration-space search method," *Computer-Aided Design,* vol. 35, no. 6, pp. 549-566, 2003.

[52] R. S. Lee and Y. H. Lin, "Development of universal environment for constructing 5-axis virtual machine tool based on modified D–H notation and OpenGL," *Robotics and Computer-Integrated Manufacturing,* vol. 26, no. 3, pp. 253-262, 2010.

[53] X. Wang, P. Zheng, Z. Wei, Y. Sun, B. Luo and Y. Li, "Development an interactive VR training for CNC machining," in *Proceedings of the ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, New York, New York, USA, 2004.

[54] R. G. Dewar, I. D. Carpenter, J. M. Ritchie and J. E. L. Simmons, "Assembly planning in a virtual environment," in *Proceedings of the Portland International Conference on Management and Technology (PICME'97),* Portland, Oregon, USA, 1997.

[55] J. M. Ritchie, J. E. L. Simmons, I. D. Carpenter and R. G. Dewar, "Using virtual reality for knowledge elicitation in a mechanical assembly planning environment," in *Proceedings of 12th*

conference of the Irish manufacturing committee, Corcaigh, Ireland, 1995.

[56] S. Jayaram, H. I. Connacher and K. W. Lyons, "Virtual assembly using virtual reality techniques," *Computer-Aided Design,* vol. 29, no. 8, pp. 575-584, 1997.

[57] C. E. Kim and J. M. Vance, "Using VPS (Voxmap PointShell) as the basis for interaction in a virtual assembly environment," in *Proceedings of the ASME Design Engineering Technical Conferences (DETC2003),*, Chicago, Illinois, USA, 2003.

[58] G. Boothroyd, "Product design for manufacture and assembly," *Computer-Aided Design,* vol. 26, no. 7, pp. 505-520, 1994.

[59] J. E. Corter, S. K. Esche, C. Chassapis, J. Ma and J. V. Nickerson, "Process and learning outcomes from remotely-operated, simulated, and hands-on student laboratories," *Computers & Education,* vol. 57, no. 3, p. 2054–2067, 2011.

[60] D. Magin and S. Kanapathipillai, "Engineering students' understanding of the role of experimentation," *European Journal of Engineering Education* , vol. 25, no. 4, pp. 351-358, 2000.

[61] B. Dalgarno, A. G. Bishop and W. Adlong, "Effectiveness of a virtual laboratory as a preparatory resource for distance education chemistry students," *Computers & Education,* vol. 53, no. 3, pp. 853-865, 2009.

[62] B. Balamuralithara and P. C. Woods, "Virtual laboratories in engineering education: the simulation lab and remote lab," *Computer Applications in Engineering Education,* vol. 17, no. 1, pp. 108-118, 2009.

[63] P. Arpaia, A. Baccigalupi, F. Cennamo and P. Daponte, "A remote measurement laboratory for educational experiments," *Measurement,* vol. 21, no. 4, pp. 157-169, 1997.

[64] Z. Nedic, J. Machotka and A. Nafalski, "Remote laboratories versus virtual and real laboratories," in *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference*, Boulder, Colorado, USA, 2003.

[65] S. Frerich, D. Kruse, M. Petermann and A. Kilzer, "Virtual labs and remote labs: practical experience

for everyone," in *Proceedings of the Global Engineering Education Conference (EDUCON)*, Istanbul, 2014.

[66] M. Stefanovic, "The objectives, architectures and effects of distance learning laboratories for industrial engineering education," *Computers & Education*, vol. 69, pp. 250-262, 2013.

[67] M. K. Jouaneh and W. J. Palm, "Control system experiments at home," in *Proceedings of the Frontier in Education Conference (FIE11)*, Rapid City, South Dakota, USA, 2011.

[68] Z. Aydogmus and O. Aydogmus, "A web-based remote access laboratory using SCADA," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 126-132, 2009.

[69] Y. Cetinceviz and R. Bayindir, "Design and implementation of an Internet based effective controlling and monitoring system with wireless fieldbus communications technologies for process automation—an experimental study," *ISA Transactions*, vol. 51, no. 3, pp. 461-470, 2012.

[70] Y. H. Elawady and A. S. Tolba, "General framework for remote laboratory access: A standarization point of view," in *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2010.

[71] A. Tekin, F. Ata and M. Gökbulut, "Remote control laboratory for DSP-controlled induction motor drives," *Computer Applications in Engineering Education*, vol. 20, no. 4, pp. 702-712, 2012.

[72] S. Hong, X. Zheng, B. Dalage, V. Kristiansen, Ø. Strøm, M. S. Shur, T. A. Fjeldly, J.-Q. Lu and T. Ytterdal, "Conducting laboratory experiments over the Internet," *IEEE Transactions on Education*, vol. 42, no. 3, pp. 180-185, 1999.

[73] D. Cazacu, "A remote Laboratory for frequency-response analysis of vibrating mechanical systems," *Procedia Technology*, vol. 12, pp. 675-680, 2014.

[74] M. Roussos, A. E. Johnson, J. Leigh, C. A. Vasilakis, C. R. Barnes and T. G. Moher, "NICE: combining constructionism, narrative and collaboration in a virtual learning environment," *Computer Graphics*, vol. 31, pp. 62-63, 1997.

[75] N. Di Blas, A. Bucciero, L. Mainetti and P. Paolini, "Multi-user virtual environments for learning:

experience and technology design," *IEEE Transactions on Learning Technologies,* vol. 5, no. 4, pp. 349-365, 2012.

[76] F. Arango, C. Chang, S. K. Esche and C. Chassapis, "A scenario for collaborative learning in virtual engineering laboratories," in *Proceedings of the 37th ASEE/IEEE Frontiers in Education Conference*, Milwaukee, Wisconsin, USA, 2007.

[77] L. Y. Joo, T. S. Yin, D. Xu, E. Thia, P. F. Chia, C. W. K. Kuah and K. . K. He, "A feasibility study using interactive commercial off-the-shelf computer gaming in upper limb rehabilitation in patients after stroke," *Journal of Rehabilitation Medicine,* vol. 42, no. 5, pp. 437-441, 2010.

[78] I. P. Acosta, "Upper limb rehabilitation of stroke patients using Kinect and computer games," in *Doctoral dissertation*, The University of Utah, 2012.

[79] H. Sin and G. Lee, "Additional virtual reality training using Xbox Kinect in stroke survivors with hemiplegia," *American Journal of Physical Medicine & Rehabilitation* , vol. 92, no. 10, pp. 871-880, 2013.

[80] Y.-J. Chang, S.-F. Chen and J.-D. Huang, "A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," *Research in Developmental Disabilities,* vol. 32, no. 6, pp. 2566-2570, 2011.

[81] L. Luna-Oliva, R. M. Ortiz-Gutiérrez, R. Cano-de la Cuerda, R. M. Piédrola, I. M. Alguacil-Diego, C. Sánchez-Camarero and M. d. C. M. Culebras, "Kinect Xbox 360 as a therapeutic modality for children with cerebral palsy in a school environment: a preliminary study," *Neuro Rehabilitation,* vol. 33, no. 4, pp. 513-521, 2013.

[82] A. Dutta, S. Chugh, A. Banerjee and A. Dutta, "Point-of-care-testing of standing posture with Wii balance board and microsoft kinect during transcranial direct current stimulation: a feasibility study," *Neuro Rehabilitation,* vol. 34, no. 4, pp. 789-798, 2014.

[83] F. Pu, S. Sun, L. Wang, Y. Li, H. Yu, Y. Yang, Y. Zhao and S. Li, "Investigation of key factors affecting the balance function of older adults," *Aging Clinical and Experimental Research,* pp. 1-9,

2014.

[84] B. Galna, D. Jackson, G. Schofield, R. McNaney, M. Webster, G. Barry, D. Mhiripiri, M. Balaam, P. Olivier and L. Rochester, "Retraining function in people with Parkinson's disease using the Microsoft kinect: game design and pilot testing," *Journal of neuroengineering and rehabilitation,* vol. 11, no. 1, pp. 60-73, 2014.

[85] J. E. Pompeu, L. A. Arduini, A. R. Botelho, M. B. F. Fonseca, S. M. A. A. Pompeu and C. Torriani-Pasin, "Feasibility, safety and outcomes of playing Kinect Adventures!™ for people with Parkinson's disease: a pilot study," *Physiotherapy,* vol. 100, no. 2, pp. 162-168, 2014.

[86] I. Parry, C. Carbullido, J. Kawada, A. Bagley, S. Sen, D. Greenhalgh and T. Palmieri, "Keeping up with video game technology: Objective analysis of Xbox Kinect™ and PlayStation 3 Move™ for use in burn rehabilitation," *Burns,* vol. 40, no. 5, pp. 852-859, 2014.

[87] L. Andrew , V. Ng and C.-H. Chan, "Gesture-Based interaction for seamless coordination of presentation aides in lecture streaming," in *Proceedings of the 8th International Conference on Information and Communication Technology in Teaching and Learning*, Hong Kong, 2013.

[88] Y.-P. Guan and P. Jing, "Human-computer interaction using pointing gesture based on an adaptive virtual touch screen," *International Journal of Signal Processing, Image Processing & Pattern Recognition,* vol. 6, no. 4, pp. 81-91, 2013.

[89] L. Cheng, Q. Sun, H. Su, Y. Cong and S. Zhao, "Design and implementation of human-robot interactive demonstration system based on Kinect," in *Proceedings of the 24th Chinese Control and Decision Conference (CCDC),*, 2012.

[90] B. D. Homer, C. K. Kinzer, J. L. Plass, S. M. Letourneau, D. Hoffman, M. Bromley, E. O. Hayward, S. Turkay and Y. Kornak, "Moved to learn: the effects of interactivity in a Kinect-based literacy game for beginning readers," *Computers & Education,* vol. 74, pp. 37-49, 2014.

[91] G. Lorenzo, J. Pomares and A. Lledó, "Inclusion of immersive virtual learning environments and visual control systems to support the learning of students with Asperger syn," *Computers &*

Education, vol. 62, pp. 88-101, 2013.

[92] L. Bartoli, C. Corradi, F. Garzotto and M. Valoriani, "Exploring motion-based touchless games for autistic children's learning," in *Proceedings of the 12th International Conference on Interaction Design and Children*, 2013.

[93] Z. Marquardt, J. Beira, N. Em, I. Paiva and S. Kox, "Super Mirror: a Kinect interface for ballet dancers," in *Proceedings of the ACM Annual Conference on Human Factors in Computing Systems*, Austin, Texas, USA, 2012.

[94] T. Schofield, J. Vines, T. Higham, E. Carter, M. Atken and A. Golding, "Trigger shift: participatory design of an augmented theatrical performance with young people," in *Proceedings of the 9th ACM Conference on Creativity & Cognition*, 2013.

[95] L. De Florian and E. Bruzzone, "Building a feature-based object description from a boundary model," *Computer-Aided Design,* vol. 21, no. 10, pp. 602-610, 1989.

[96] C. Li and K. C. Hui, "Feature recognition by template matching," *Computers & Graphics,* vol. 24, no. 4, pp. 569-582, 2000.

[97] K. Ulrich, "The role of product architecture in the manufacturing firm," *Research Policy,* vol. 24, no. 3, pp. 419-440, 1995.

[98] M. N. K. Boulos, B. J. Blanchard, C. Walker, J. Montero, A. Tripathy and R. Gutierrez-Osuna, "Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation," *International Journal of Health Geographics,* vol. 10, no. 1, pp. 1-14, 2011.

[99] R. Francese, I. Passero and G. Tortora, "Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, Naples, Italy, 2012.

[100] H. Jack, "Part storage and separation," in *Engineering Design, Planning, and Management* , Kidlington, UK, Academic Press is an imprint of Elsevier, 2013, p. 427.

Vita

Education:

Stevens Institute of Technology, Hoboken, New Jersey

- Doctor of Philosophy in Mechanical Engineering, Expected December 2016
- Master of Engineering in Mechanical Engineering, May 2012

Tianjin University, Tianjin, China

- Bachelor of Engineering in Measuring and Control Technology and Instruments, July 2009
- Bachelor of Engineering in Computer Science and Technology, July 2009

Journal publications:

1. Chang, Y., Aziz, E.-S., Zhang, Z., Zhang, M. & Esche, S. K. (2015). Usability evaluation of a virtual educational laboratory platform. Accepted for publication *Computers in Education Journal*.
2. Aziz, E.-S., Chang, Y., Esche, S. K. & Chassapis, C. (2015). Virtual mechanical assembly training based on a 3D game engine. In *Computer-Aided Design and Applications*, Vol. 12, No. 2, pp. 119-134.
3. Chang, Y., Aziz, E.-S., Esche, S.K, Chassapis, C. (2013). A framework for developing collaborative training environments for assembling. *Computers in Education Journal*, Vol. 23, No. 4, pp. 44-59
4. Aziz, E.-S., Chang, Y., Esche, S. K., & Chassapis, C. (2013). A multi-user virtual laboratory environment for gear train design. *Computer Applications in Engineering Education*, Vol. 22, No. 4, pp. 788 – 802

5. Chang, Y., Aziz, E.-S., Esche, S. K. & Chassapis, C. (2012). A game-based laboratory for gear design. *Computers in Education Journal*, Vol. 22, No. 1, pp. 21-31.

6. Yu, Y., Zhang, H., Wang, Z., & Chang, Y. (2010). Deep-hole inner diameter measuring system based on non-contact capacitance sensor. *Transactions of Tianjin University*, Vol. 16, pp. 447-451.

Conference proceedings:

7. Chang, Y., Aziz, E.-S., Esche, S. K. & Chassapis, C. (2014). A platform for mechanical assembly education using the Microsoft Kinect. *ASME International Mechanical Engineering Conference & Exposition IMECE'14*, Montreal, Quebec, Canada, November 14 - 20, 2014.

8. Zhang, Z., Zhang, M., Chang, Y., Esche, S. K. & Chassapis, C. (2014). An efficient method for creating virtual spaces for virtual reality," *ASME International Mechanical Engineering Conference & Exposition IMECE'14*, Montreal, Quebec, Canada, November 14 – 20, 2014.

9. Zhang, Z., Zhang, M., Chang, Y., Aziz, E.-S., Esche, S. K. & Chassapis, C. (2013). Real-time 3D model Reconstruction Using Kinect for A Game-Based Virtual Laboratory. *ASME International Mechanical Engineering Conference & Exposition IMECE'13*, San Diego, California, November 13 – 21, 2013.

10. Aziz, E.-S., Chang, Y., Esche, S. K. & Chassapis, C. (2012). Capturing assembly constraints of experimental setups in a virtual laboratory environment. *ASME International Mechanical Engineering Conference & Exposition IMECE'12*, Houston, Texas, November 9 - 15, 2012.

11. Aziz, E. S., Corter, J. E., Chang, Y., Esche, S. K., & Chassapis, C. (2012). Evaluation of the learning effectiveness of game-based and hands-on gear train laboratories. *42nd ASEE/IEEE Frontiers in Education Conference FIE'12*, Seattle, Washington, October 3 - 6, 2012.

12. Chang, Y., Aziz, E. -S., Esche, S. K., & Chassapis, C. (2011). Overcoming the Limitations of Current Online Laboratory Systems Using Game-Based Virtual Environments. *ASME*

International Mechanical Engineering Conference & Exposition IMECE'11, Denver, Colorado, November 11 - 17, 2011.

13. Chang, Y., Aziz, E. -S., Esche, S. K., & Chassapis, C. (2011). Assessment of the pilot implementation of a game-based gear design laboratory. *41st ASEE/IEEE Frontiers in Education Conference FIE'11*, Rapid City, South Dakota, October 12 - 15, 2011.
14. Aziz, E. -S., Chang, Y., Tumkor, S., Esche, S. K., & Chassapis, C. (2010). Adapting computer game technology to support engineering laboratories. *ASME International Mechanical Engineering Conference & Exposition, IMECE'10,* Vancouver, British Columbia, Canada, June 26 - 29, 2011.

Experiences:

Mechanical Engineering Department, Stevens Institute of Technology, Hoboken, New Jersey

- Research Assistant, Sept 2009-June, 2016
- Teaching Assistant, Sept 2012-June, 2016

National Institute of Metrology, Beijing, China

- Research Assistant Jan 2009-July 2009

Skills:

- Computer language: C, C++, C#, LabVIEW, MATLAB script, Lua script
- Design/simulation software: SolidWorks, Pro/Engineer, 3ds Max, Arena, Ansys, AutoCAD
- Computer graphics/gaming: OpenGL, OGRE Rendering, Source SDK, Microsoft Kinect SDK, OpenCV
- Imbedded system: Arduino MPU, Intel 8051 Series, Assembly Language
- Database: Microsoft Access, Microsoft SQL Server

- Languages: Chinese, English

Presentations:

- Usability evaluation of a virtual educational laboratory platform, on 122nd ASEE Annual Conference & Exposition, Seattle, Washington, June 16, 2015
- A platform for mechanical assembly education using the Microsoft Kinect, on ASME International Mechanical Engineering Conference & Exposition, Montreal, Quebec, Canada, November 17, 2014

Services:

- Reviewer for 2015 ASEE Annual Conference and Exposition
- Reviewer for 2013 ASEE Annual Conference and Exposition
- Reviewer for 2013 Frontiers in Education Conference