A Framework for Developing Collaborative Training Environments for Assembling

Yizhe Chang, Stevens Institute of Technology

Yizhe Chang is currently a Ph.D. student in Mechanical Engineering Department, Stevens Institute of Technology. He received his B.Eng. from Tianjin University, Tianjin, China. His current research topics include virtual environment for assembly simulation and collaborative system for engineering education.

Dr. El-Sayed S. Aziz, Stevens Institute of Technology (SES)

Dr. El-Sayed Aziz holds a faculty position as associate professor in the Production Engineering and Mechanical Design Department at Faculty of Engineering, Mansoura University, Egypt. Currently, he is working as a research scientist at Stevens Institute of Technology in Hoboken, New Jersey. He received B.S. and M.S. in Mechanical Engineering from Mansoura University in Egypt in 1991 and a Ph.D. in Mechanical Engineering from Stevens Institute of Technology in 2003. His research interests include knowledge-based engineering systems, computer-integrated design and manufacturing, Finite Element Analysis, software development and applications as well as remote and virtual laboratories.

Dr. Sven K. Esche, Stevens Institute of Technology
Dr. Constantin Chassapis, Stevens Institute of Technology (SES)

A Framework for Developing Collaborative Training Environments for Assembling

Abstract

State-of-the-art 3D video games can provide their users with a near-real experience from visual, audio and interactivity perspectives. Numerous efforts have been made to take advantage of these favorable characteristics for educational purposes. The majority of these projects have focused either on the reconstruction of certain scenarios, such as fire emergency response training, driver or pilot training, medical training, and military tactics training, etc., or on the realistic simulation of real environments, such as virtual museum tours and virtual architectural tours. However, only recently have attempts been made to develop virtual engineering training environments since those require the assembly of different types of individual components into potentially complex systems and involve interactions between multiple participants and models of physical devices.

In this article, a framework for developing collaborative environments for mechanical assembly training based on a commercial computer game engine will be introduced. Such environments not only maintain the feel of immersion of a 3D game, but they also emphasize realistic physical device management and control. In this context, the trainees are enabled to manage a large variety of mechanical components, assemble them according to engineering requirements or potentially disassemble them. Also, the instructors can monitor the entire assembly process and automatically collect performance statistics. Therefore, such environments have the potential to become valuable workforce development tools that enable their users to acquire practical skills for assembling a variety of electro-mechanical systems for the purpose of manufacturing, maintenance and repair.

1. Introduction

For some time, educators from different areas have been designing video games for improving teaching or training practice. Such video games are required to integrate instruction strategies and ludic activities to benefit educational goals[1].This idea has been implemented for different functional purposes. Some video games reconstruct certain scenarios while others simulate real environments. As examples for the former type, there are reports of using video games for fire drill[2,3], driver[4] and pilot training[5], medical training[6,7], and military skills and tactics training[8]. Examples for the latter type include virtual museums[9], virtual architectural tours[10,11] and virtual classrooms[12]. Recently, classroom multiplayer presidential games are being introduced to replace slideshows by bringing a virtual world directly into classrooms.[13,14]

However, for virtual engineering training environments, especially with scenarios of mechanical assembly, the attempts made so far are limited.[15] This is because on one hand, a mechanical assembly process involves the integration of various types of individual components into a complex system, which may be complicated. On the other hand, for such an assembly training environment, other factors such as the involvement of multiple participants who play different roles, the mechanical modeling and the interactions between the participants and mechanical components, also need to be considered.

A successful implementation of a virtual mechanical assembly environment may draw on the experiences from both the contemporary video game industry and modern computer-aided design/manufacturing research.

1.1. Game-engines as development platforms for virtual environments

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, 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 artificial intelligence engines. In online multiplayer games, communication modules are also included to enable the 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, weapons, etc. The development of this layer requires combined efforts of computer technicians and, more importantly, artists.

It is a convenient way for virtual mechanical assembly developers to utilize the resources provided by the game engine layer in order to devise the game play layer, while concentrating on composing game scenarios, planning the story line, assigning avatars for players and non-player characters (NPC) and modeling necessary gadgets. Although most game engines are designed for entertainment purposes, the basic functions for graphics, physics simulation and story plots are capable of supporting the design of educational video games. In addition, nowadays, many game engines are not game-specific but rather 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 developments 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) for game developers to build their own games.

1.2. Modeling of mechanical assemblies

As previously stated, mechanical assemblies may be complex systems. Developing a virtual assembly training environment poses a number of problems, such as how to model mechanical parts and the connections between them, how to plan assembly sequences and how to analyze the kinematics of the assemblies. Although they have some functions that partially cover these problems, game engines are designed for entertainment and thus emphasize the entertainment factors such as rendering effects and speed rather than features that facilitate mechanical assemblies. In order to build a virtual environment for mechanical assembly training, it is beneficial to draw on experiences from the computer aided design/manufacturing (CAD/CAM) area. The modeling of mechanical assemblies has been widely researched for CAD/CAM and design for assembly purposes.

Generally, a mechanical assembly consists of individual parts, which are further connected by different types of associations. The concepts of feature-based part modeling and constraint-based assembly modeling are well developed. In feature-based part modeling, a part is modeled by 'features'. The features include functional features (e.g. assembly features or physical features) and form features (e.g. geometric features). Usually, when mentioning 'feature', the form feature is meant.[16] A form feature describes the geometric configurations formed on the surface, edge or corner of a mechanical part[17], or more broadly, a generic shape that carries certain engineering meaning.[18] Most state-of-the-art CAD software is developed based on the form feature concept for users to model the geometry of parts. In 1994, the International Organization for Standardization (ISO) initiated the Standard ISO 10303, colloquially STandards for Exchange of Product (STEP), which regulates the mechanical component model data to facilitate the communication between different CAD software.[19]

Researchers further proposed mechanical assembly models consisting of feature-based parts. Assembly features can be defined upon form features. Form features that are able to match two or more parts can form 'feature associations', for instance, a pin-hole association. These feature associations impose constraints to the parts which suppress certain kinematical degrees of freedom. For instance, the pin-hole association imposes a joint constraint that suppresses five degrees of freedom, leaving only one rotational degree of freedom.

How can constraints that are imposed inside of an assembly be simulated graphically? Constraint solvers are designed to tackle this problem. With the help of artificial intelligence techniques, various types of constraint solvers have been developed, for instance local propagation solvers, finite-domain solvers, numeric solvers, symbolic solvers, geometric solvers and domain-specific solvers, etc.[20] Each of them have advantages and disadvantages. For instance, local propagation solvers, which are most commonly used, are efficient, easily customizable and robust, but they cannot handle cyclic constraints and non-equation constraints.

CAD research has also explored the concept of collaborative assembly and design on the basis of graphics and Internet technology. The idea of collaborative assembly and design is to facilitate the product development management, which spans the whole lifecycle of the product and involves the cooperation of diverse organizations using diverse CAD systems.[21] The ultimate goal of collaborative assembly is to build a platform that allows users with different tasks to share their design concepts, design geometries (i.e. 3D part models) and technical details.[22]

In the Chapter 2, the details of a game-based environment for assembly training that incorporates concepts developed by CAD/CAM research will be introduced.

2. Virtual Assembly Training Environment

The 'Source' game engine is one of the commercial game engines that is widely distributed and used to develop multiplayer 3D games such as 'Half-Life 2' and 'Counter Strike'. With the help of the capabilities of the game engine with respect to rendering, physics, audio, artificial intelligence, networking, etc., a physics sandbox game modification, commonly known as 'Garry's Mod', was developed by Team Garry. 'Garry's Mod' allows game players to exert their creativity by building and simulating their own gadgets. What is more important is that 'Garry's Mod' allows game developers to use 'Lua' scripts, an easy-to-learn scripting language, to call the physics or rendering functions inside of the game engine. By coding in 'Garry's Mod', the mechanical assemblies can be prototyped.

‘Garry’s Mod’ also adopts the model import methods that are used by the ‘Source’ game engine. Mechanical parts, which are modeled in 3ds-Max or CAD modeling software such as SolidWorks, can be moved into ‘Garry’s Mod’ as vertex-based solid models.[23] Their geometries, dimensions and textures can be edited first in the above modeling software and then directly transferred to ‘Garry’s Mod’ models. Game developers can script these models so as to endow them with properties such as mass, or even let them exhibit special behaviors, such as requiring a cube to roll like a ball by equipping it with a spherical bounding volume. Models that are scripted are called ‘scripted entities’ (SENTs) in ‘Garry’s Mod’. They are the fundamental elements of the mechanical assemblies in the application described here.

Mechanical models alone are only parts of the virtual environment. Broadly speaking, a virtual environment must have a background such as rooms, furniture, decorations and sometimes non-player characteristics (NPC) to strengthen the environment’s immersive characteristics. The ‘Hammer’ map editor, which is part of the ‘Source’ SDK, can resolve this issue. The map editor allows game developers to create their own game maps (in our context, the assembly factory or assembly training laboratory).

The game engine layer for the virtual assembly environment is laid out by ‘Lua’ scripts to flexibly code the mechanical parts and assemblies, by modeling software to construct precise mechanical parts, and by the map editor to build a customized and professional environment. Figure 1 shows the software that was applied to develop the virtual assembly environment.

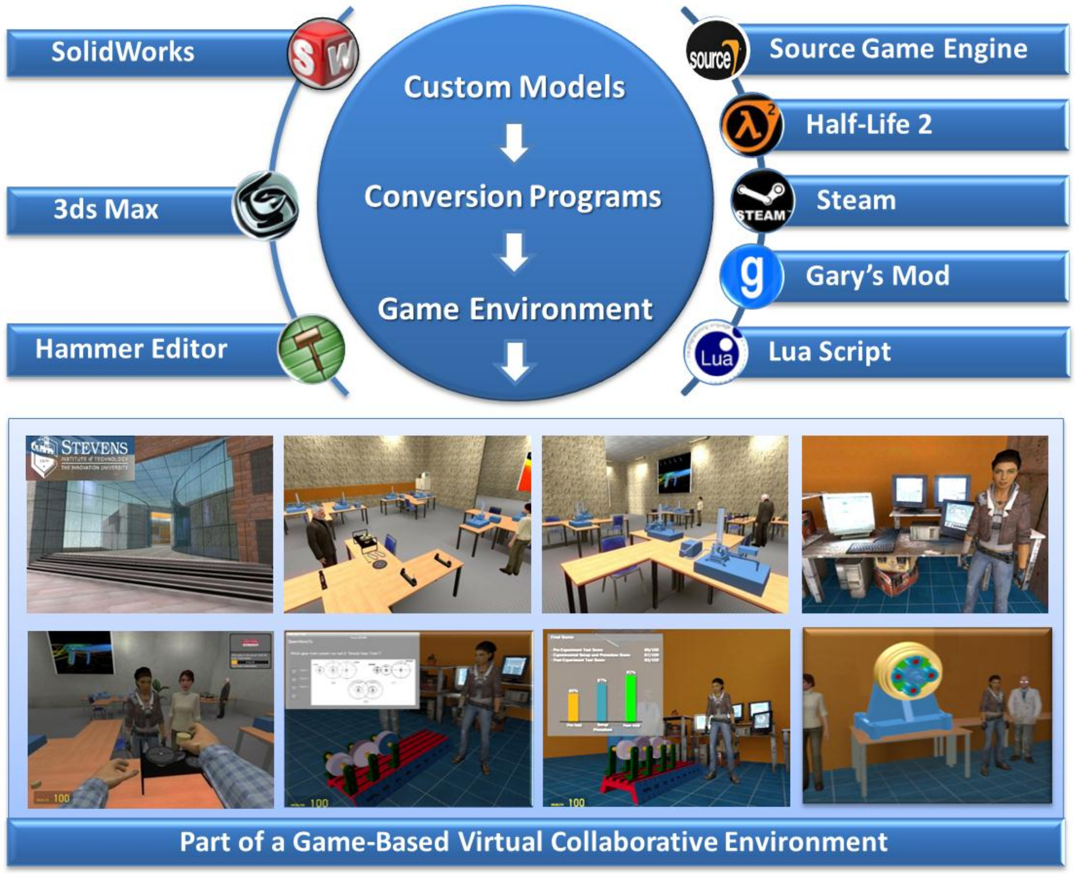

Figure 1: Overall process for developing a game-based virtual collaborative environment

3. Assembly Model

An assembly should at least contain the following information:[24]

- Mechanical parts. Mechanical parts are solid 3D geometries which cannot be further split (e.g. gears, shafts or bearings).
- Assembly configuration. An assembly configuration represents the hierarchical relationship between the components (assemblies, sub-assemblies and parts).
- Part association. The part association provides detailed information for joints between the pairs of connected parts. In feature based assembly, part association is implemented on the basis of feature association. The associations can be classified as:
  a. Moveable connections where the pair is connected and movable (e.g. shaft-bearing joints, slider-guideway joints, gear joints, etc.).
  b. Fixed connection where all relative degrees of freedom are suppressed (e.g. bolted or welded connections).

With these guidelines, a virtual assembly environment can be built. However, there are still many concepts to be clarified, such as features, feature associations, hierarchical assembly structure, assembly sequence modeling and how to integrate these concepts with scripted entities (SENTs) as mentioned in Section 2.

In order to illustrate these issues, a planetary gear train is described below. Because planetary gear trains are characterized by high power transmission efficiency and low space requirements, they are widely used in industry and thus became an important part of the knowledge for mechanical trainees. Planetary gear trains consist of a central sun gear that is engaged with the surrounding planet gears, which in turn are held by a carrier that constrains the planets on an orbit. A ring gear with internal teeth meshes with each of the planet gears. Different gear ratios between the input and output are achieved by different combinations of locked and unlocked gears. Although planetary gear trains are fairly simple assemblies, they cover the main concepts introduced in this article. Figure 2 depicts an exploded view of a planetary gear train.

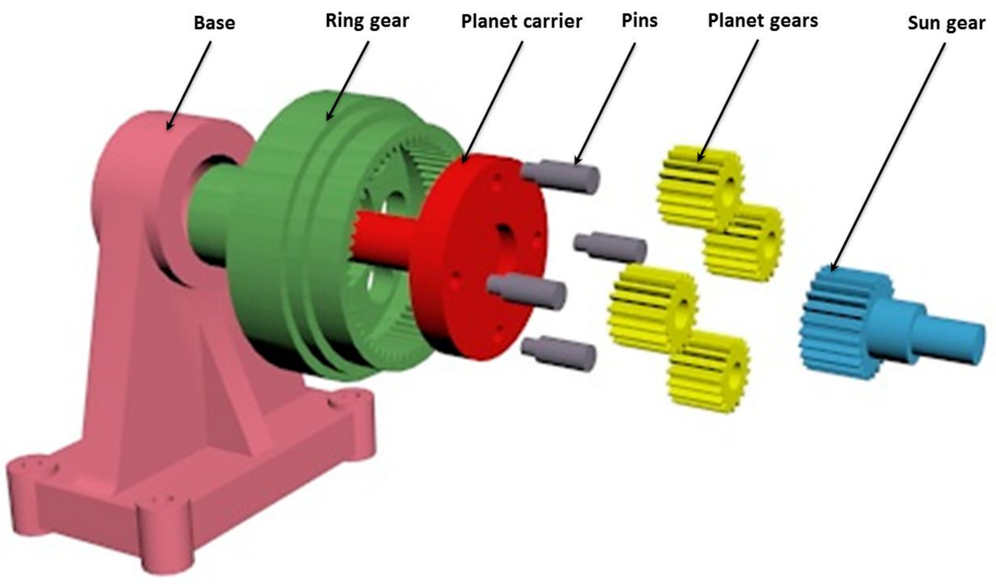

Figure 2: Exploded view of planetary gear train assembly

3.1. Features and feature slots

Feature-based modeling is common in modern CAD software. In commercial CAD software such as SolidWorks or Pro/Engineer, parts are modeled by parametric features. For instance, if a hole is expected to be drilled into a cube, the user needs to define the position of the hole, add a hole feature there and then assign values for the diameter and the depth of the hole. After the hole has been defined, a hole feature is inserted in the feature tree. Later, the user can modify the feature or delete it in a convenient fashion. Contrary to the approach in CAD software, in ‘Garry’s Mod’ (and in most game engines), models are merely represented by triangular planes, which are actually described and stored by their vertices. The concept of features does not exist at all. For instance, a hole and a cylinder are not different as far as their models are concerned, except that the locations of their vertices differ.

In ‘Garry’s Mod’, design features have to be redefined based on the vertex geometry in the form of feature slots. Since the goal here is only to model the assembly-related aspects, the features are restricted to those characteristics that are meaningful for the assembly while neglecting other irrelevant characteristics. For instance, the web on the base part (see Figure 3) does not have a specific function in the assembly and therefore it is not necessary to define it as a feature, while the cylindrical hole can match a shaft so that a feature will be defined upon it.

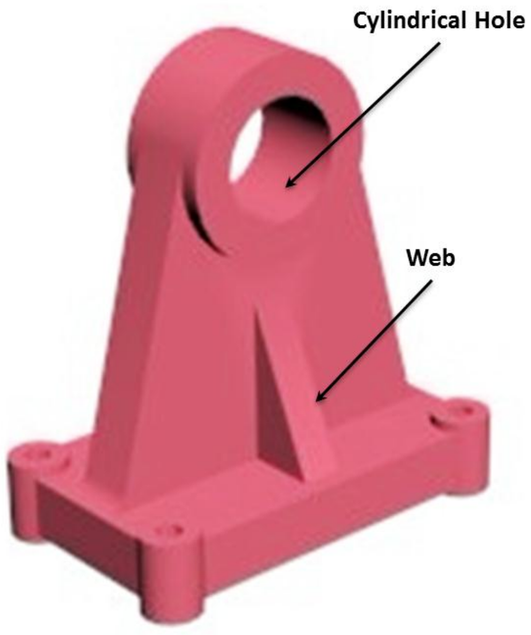

Figure 3: An example of defining features on a part.

A feature slot is a data package that describes a specific feature on a part. Each slot holds attributes of that feature, such as the type of the feature, the position of the feature, etc. Table 1 lists the major attributes of a feature slot. A ‘feature_availability’ attribute was devised to prevent one feature slot from establishing up feature associations with more than one feature slot simultaneously.

Table 1 Major attributes of feature slots

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'

Figure 4 shows an example for the application of the feature slot. In a planet gear train, the planet carrier has feature associations with the sun gear, the ring gear and the pins, and thus it has multiple matching feature slots with these parts. In order to mate with the ring gear (which has a cylindrical hole feature), feature slot (1) (which is a cylinder feature) is assigned to the external surface of the cylinder. In addition, this feature slot includes the outer diameter d_1 of the cylinder to be mated with the inner diameter of the cylindrical hole in the ring gear. Likewise, slots (3)-(6) are assigned to the cylindrical hole features with diameter d_3 located at their respective local positions on the planet carrier. Analogously, slot (2) is assigned to the interior surface of the cylindrical hole feature of diameter d_2.

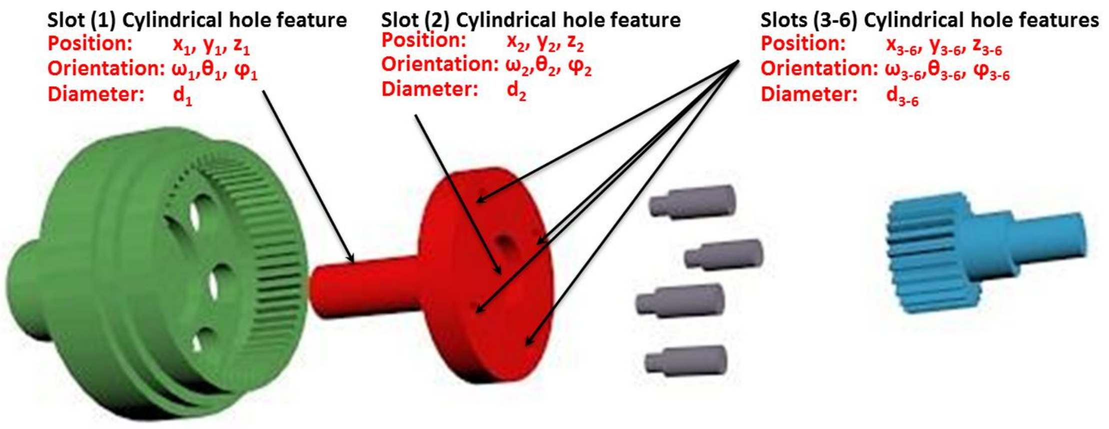
Slot (1) Cylindrical hole feature
Position: x_1, y_1, z_1
Orientation: $\omega_1, \theta_1, \varphi_1$
Diameter: d_1
Slot (2) Cylindrical hole feature
Position: x_2, y_2, z_2
Orientation: $\omega_2, \theta_2, \varphi_2$
Diameter: d_2
Slots (3-6) Cylindrical hole features
Position: $x_{3-6}, y_{3-6}, z_{3-6}$
Orientation: $\omega_{3-6}, \theta_{3-6}, \varphi_{3-6}$
Diameter: d_{3-6}

Figure 4: Features and feature slots of planet carrier

3.2. Feature association

In feature-based assembly, the paired parts of an assembly form part associations on the basis of feature associations. A feature association is a constraint that suppresses one or multiple relative degrees of freedom. Analogously to part associations, feature associations are classified into moveable or fixed ones. The former remove 5 or fewer relative degrees of freedom (for instance, revolute and ball joints eliminate 5 and 3 degrees of freedom, respectively) while the latter eliminate all 6 relative degrees of freedom (e.g. welding constraints).

Feature associations are constructed on the basis of feature slots. Two paired parts that contain matching feature slots can be linked only if both feature slots are in the non-occupied state. Once the joint has been established, the kinematics of the paired parts is regulated by the feature association.

In 'Garry's Mod', fundamental feature associations are defined in the form of ready-to-call functions as listed in Table 2. Therefore, technically implementing a feature association of one of the common types is straightforward. Table 2 lists the common feature associations that are supported by 'Garry's Mod', most of which have an alternative name in the game.

Table 2: Constraints used in 'Garry's Mod'

Constraint in 'Garry's Mod'	**Feature association in mechanical assembly**	**Description**
Axis constraint	Revolute joint	Joins two parts with an axis about which they can spin freely; their relative positions in axial direction are fixed
Ball socket	Spherical joint	Joins two parts with same center point about which they can rotate freely in all directions
Elastic constraint	Elastic joint	Connects two parts with a spring-like rope that, when compressed or stretched, will try to resume its original length
Slider	Prismatic joint	Creates a path along a straight line on a part that a matching part can travel along
Weld	Fixed joint	Joins two parts such that afterwards they can no longer be moved relative to each other

3.3. Hierarchical assembly

In the real world, when joining two parts, there may or may not be an implicit understanding of which part moves and which part is fixed. Contrary to this, in a game environment, the fixed part must be explicitly pre-determined. For instance, when one says 'join a desk and a book', the default meaning would be to move the book onto the fixed desk, rather than to move the desk to let it be beneath a book that is 'floating in the air'. In CAD software, there is no such issue as there being no reference like a floor or walls. In order to put the book on the desk in a CAD software, one could move the desk under the book to form a 'sub-assembly' and then maybe move the sub-subassembly under another book. Finally, this would result in a desk with two books piled up on it. In a virtual environment, such a movement would look strange since it would result in the desk not standing on the floor. Of course, in 'Garry's Mod' the desk would eventually fall back to the floor because of the effect of gravity modeled by the physics engine. However, this process would be rather confusing to the trainees.

In order to avoid this confusion, a hierarchical part relationship is proposed here. For modeling an assembly process in the virtual environment, all types of parts are assigned an integer rank so that when a low-rank part is ordered to be associated with a high-rank part, the low-rank part always moves toward the high-rank part. In practice, rank 0 is the highest rank, and the larger its rank number is, the lower rank a part has.

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. Figure 5 shows the hierarchy of the planet gear train. 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 the 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 may not be suitable for another sequence. For example, the preferred sequence of assembling the planet gear and pin is, first, inserting the pin into the carrier and then mounting the planet gears onto the pins. Therefore, since the rank of the carrier is 2, the rank of the pin should be 3 and that of the planet gears should be 4. However, it is also acceptable to first connect the planet gears and pins to create planet-pin sub-assemblies and then insert these sub-assemblies into the holes in the planet carrier. In this situation, since the pins have rank 3 and the planet gears have rank 4, during forming 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 again not be suitable for the first assembly sequence.

In order to resolve this issue, the concept of the rank of sub-assemblies is 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, the rank of the carrier-pin sub-assembly takes on the rank of the carrier, which is 2. Since the planet gear has rank 4 and the sub-assembly has rank 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, the pin would move to the planet gear to form the sub-assembly since it is the lower-ranked part. 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. However, it is always better to assign a new rank to it. In this case, as shown in Figure 5, the sun gear is assigned to be in rank 5, a rank that is different from others.

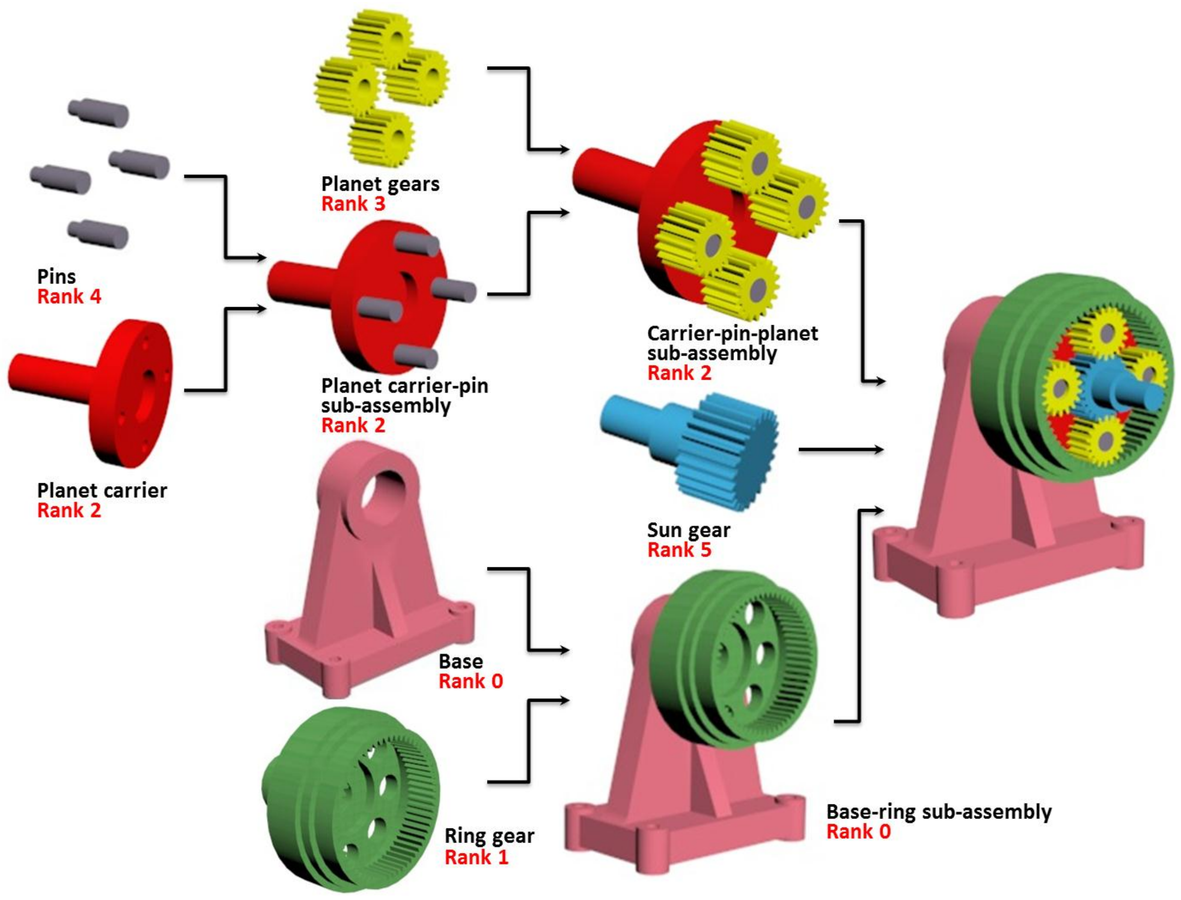

Figure 5: Hierarchy of gear train assembly

4. Operations in Game-based Environment

4.1. Assembly operation

In CAD software, the operator can order an assembly by choosing exact matching surfaces. In the virtual training environment, it is impractical to let the trainees pick the matching surfaces since, on the one hand, clicking on a specific surface is hard to implement technically in the game-based environment, and on the other hand, the game-based environment is intended to provide the trainees the immersive feeling that they are grabbing those parts, but choosing the surfaces and matching them cannot provide such reality. Most game engines provide a method for simulating grabbing activities. In the virtual environment based on 'Garry's Mod', the trainees can hold or grab something by a tool called 'physics gun' as shown in Figure 7 (left), which provides the feeling of grabbing and the handling of which is easy to learn.

In the real world, when one wants to assemble two parts (i.e. establish a feature association between two parts), say, insert a pin into the hole of a planet gear and build a revolute joint between them, one can grab the pin and insert it into the planet gear directly. In the game-based environment, the same procedure is followed. However, in the real world, one can and must align the pin exactly with the central axis of the hole. Unfortunately, in the game-based environment, it is almost impossible to simulate such an operation with a mouse and a keyboard, even if the trainee is experienced with the controls of the game.

Therefore, the operation of assembling two parts was simplified in the virtual environment. Trainees can request the assembly of two parts by grabbing one part and letting it collide with the other part. Then, the virtual environment processes this request and subsequently either rejects or accepts the request. In the former case, no assembly is made while in the latter case, a constraint is imposed between the two parts.

This process has three major steps:

- Identify all pairs of feature slots that are qualified for being assembled
- Determine the exact assembly position
- Establish the feature association and impose kinematic constraints on both parts

Figure 6 illustrates the detailed processes of establishing a feature association by the virtual environment after a trainee requests the assembly of two parts by letting them collide.

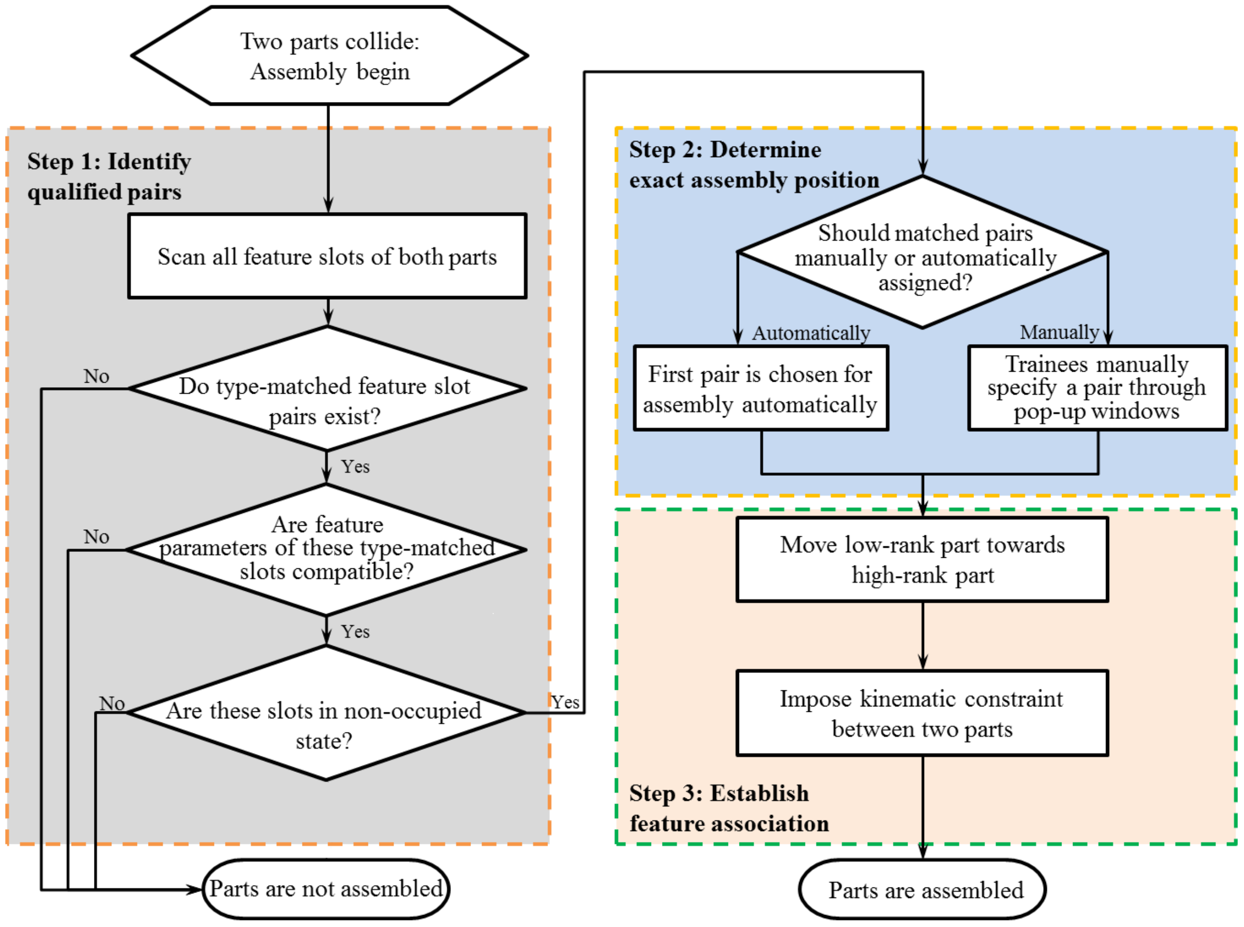
Two parts collide: Assembly begin
Step 1: Identify qualified pairs
Scan all feature slots of both parts
Do type-matched feature slot pairs exist?
Are feature parameters of these type-matched slots compatible?
Are these slots in non-occupied state?
No
Yes
Parts are not assembled
Step 2: Determine exact assembly position
Should matched pairs manually or automatically assigned?
Automatically
Manually
First pair is chosen for assembly automatically
Trainees manually specify a pair through pop-up windows
Move low-rank part towards high-rank part
Impose kinematic constraint between two parts
Step 3: Establish feature association
Parts are assembled

Figure 6 Flowchart of establishing a feature association after a trainee lets two mechanical parts collide

In the first step, three tests are conducted. The first test is to find matching feature types. During this test, all feature slots on both parts are searched and those feature slots that have slots on the other part with matching feature type are kept while the rest are excluded from further consideration. If there are no matching slots, the assembly process is terminated automatically and no feature association is built between these two parts. The second test is to check whether the detailed parameters of these type-matched feature slots are compatible. For instance, if a pair of type-matched feature slots that consists of a 5 mm diameter cylindrical hole and a 10 mm diameter cylinder, this pair would fail in this test. The last test is to check the occupancy of these type- and parameter-matched slots. In order to qualify for a feature association, neither of the slots can be in occupied state. The matching slot pairs that have passed all three tests are qualified pairs and will be passed to the second step.

In the second step, the exact slots for assembly are determined. There are situations where there is more than one qualified slot pair (for instance, the planet carrier has four identical feature slots for pins). Once there is more than one matching pair, one of the following options can be chosen in order to decide which qualified pair will be used for establishing the feature association:

- Automatically establish the feature association between the first matching pair. Since all feature slots are numbered and thus sorted, the assembly can always start with the first available feature slot. This is the easiest way for trainees.
- Use a pop-up window to ask the trainee to pick a feature slot. By doing so, the trainee can freely choose which feature slot should be used.

Once a pair has been chosen (either automatically or manually), the association is ready to be established. The last step is establishing the association. First, the positions of the parts are adjusted in order to move them in the correct assembly position and orientation. For example, the pin moves exactly into the selected pin hole of the carrier with correct position and orientation. As mentioned in Section 3, only the lower-rank part is translated and rotated to comply with the position and orientation of the higher-rank part. Finally, the feature association is established.

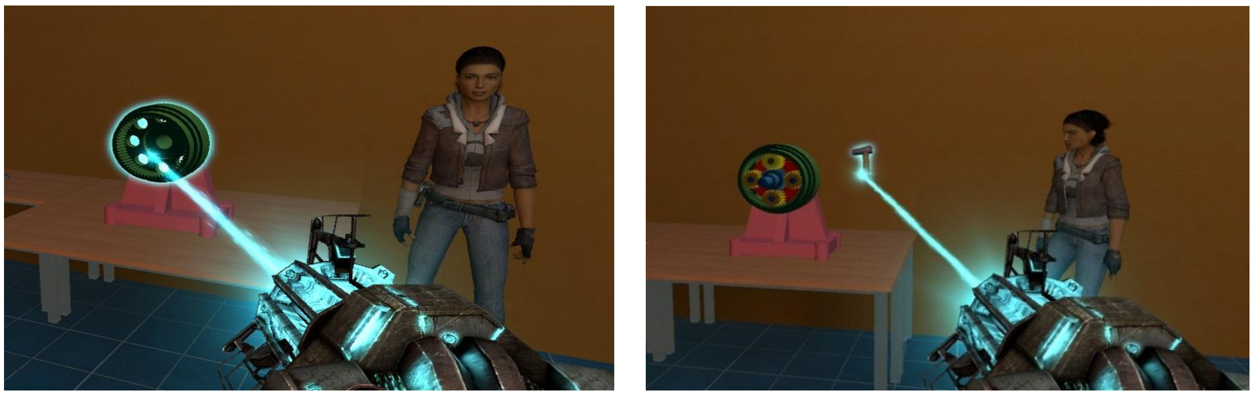

Figure 7: Assembly and disassembly operation in grabbing a ring gear by the ’physics gun’ and mounting it on the base (left) and using a ‘hammer’ (disassembly tool) to signal a disassembly (right)

4.2. Disassembly operation

When the trainee has mistakenly assembled two parts, there must be a method for disassembling them. The game-based virtual training environment should provide a disassembly function. The disassembly procedure is similar to the assembly procedure, except that the feature slots are freed up and the constraints between mating parts are removed.

As in the case of assembly operations, there must be a method for the trainees to notify the computer that a disassembly is desired. Unlike in CAD software where the operator can remove any constraint by simply deleting it from the assembly tree, in the game-based virtual environment, the disassembly is initiated by a trainee using a special tool, a ‘hammer tool’ in the current virtual environment, to select the parts to be disassembled (see the right picture of Figure 7).

As in the assembly process, the disassembly occurs after the collision of the hammer and the part that is being disassembled, except that the process of disassembling is much simpler than the assembly process. First, all feature slots on the part selected for disassembly are automatically scanned and the occupied feature slots are identified. Then, all constraints related to the occupied feature slots are removed and these feature slots are reclassified as non-occupied.

5. Application and evaluation of the framework in mechanical educational laboratory

This framework has been applied in developing a planet gear train laboratory, which is a part of the junior-level course “Mechanisms and Machine Dynamics”. In the development of the laboratory, the assembly rules described in Section 3 were applied and all mechanical parts, such as planet gears, sun gears, ring gears, etc. described in that section were modeled and coded. Students can use the operations described in Section 4 to assemble or disassemble planet gear trains. By performing the laboratory exercises, students can enhance their understanding of planet gear trains.

The laboratory provides three sets (each with different dimensions) of sun gears, ring gears, planet gears and planet carriers for the students to choose from. Only three combinations of dimensions are valid choices for assembling a planet gear train. These combinations are not given to the students. Instead, the students are required to find them on their own. The students can either choose correct combinations by analytical means based on gear train theory or by trial and error by attempting to assemble different combinations in the virtual environment.

This laboratory exercise was administered to 35 mechanical engineering students, who were divided into groups of 2 or 3. The detailed procedure of this laboratory exercise and the learning effectiveness of the laboratory exercise were discussed in detail previously[25].

Here, the focus is on the process of the students performing the assembly in the virtual environment. From the student activities in the laboratory, it was observed that all groups were able to complete the assembly (with the help of a teaching assistant). Also, it was noticed that most groups followed the (desired by the instructor) procedure of choosing a suitable combination of dimensions by analysis first and then fixing any possible analysis problems by experimenting in the virtual environment. If their analysis was correct, the students were able to complete the assembly with one trial. However, those who did not analyze the dimensions at all before attempting the assembly had to try several combinations in the virtual environment. In the worst case, one student group could theoretically have to try as many as 27 different combinations in order to build a viable planet gear train. A histogram illustrating the number of trials is shown in Figure 8.

From Figure 8, it can be seen that among 15 groups, 6 groups (40%) were able to complete the assembly with one trial. This fact shows that these students fully understood the gear train theory and performed a correct gear train analysis prior to assembling their design in the virtual environment. Two groups tried 7 times, which is still much less than trying to assemble gear trains randomly. This indicates that they had done analysis before or during the assembly. This also shows that through trials, the students were able to gradually test combinations of dimensions in the virtual environment, which eventually led them to a viable combination.

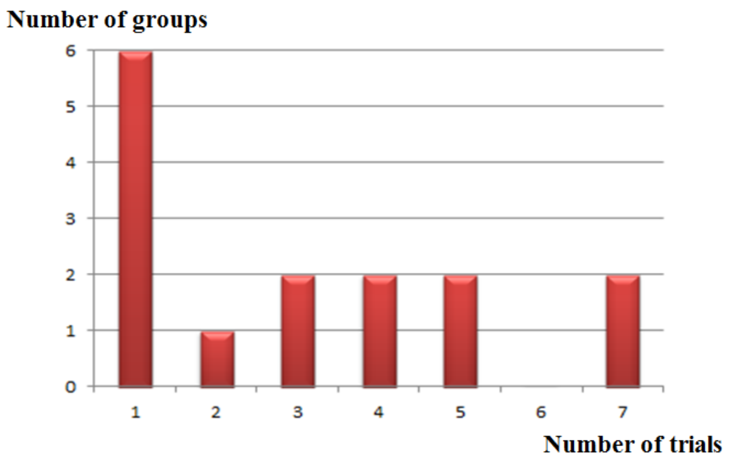

Figure 8: Histogram of number of groups with respect to number of trials made by students

6. Conclusion

This paper describes a framework for mechanical assembly training in a game-based collaborative virtual environment. The requirements and theoretical foundations are analyzed and the virtual environment is presented. Several concepts of a mechanical assembly in the game-based environment are addressed, which include design features, feature slots, feature associations and hierarchical assemblies, wherein the features and feature slots are the fundamental elements of the resulting assemblies. Finally, assembly and disassembly operations are discussed.

Based on this framework, a virtual laboratory environment for mechanical assembly was developed and laboratory experiments were conducted by mechanical engineering students. Judging from observation of the students' performance in the laboratory, the virtual assembly environment shows promise as an assembly training tool. Furthermore, this framework has the potential for being applied for industry-level training and education.

References

1 Amory, A. (2007). Game object model version II: a theoretical framework for educational game development. *Educational Technology Research and Development*, *55*(1), 51-77.

2 Smith, S. P. & Trenholme, D. (2009). Rapid prototyping a virtual fire drill environment using computer game technology. Fire Safety Journal, *44*(4), 559-569.

3 Dugdale, J., Pavard, B., Pallamin, N., el Jed, M. & Maugan, L. (2004). Emergency fire incident training in a virtual world. *Proceedings of the International Workshop on Information Systems for Crisis Response and Management 2004.*

4 Bayarri, S., Fernandez, M. & Perez, M. (1996). Virtual reality for driving simulation. *Communications of the ACM*, *39*(5), 72-76.

5 O'Neil, H. F., Mayer, R. E., Herl, H. E., Niemi, C., Olin, K. & Thurman, R. A. (2000). Instructional strategies for virtual aviation training environments. *Aircrew Training and Assessment*, 105-130.

6 Lee, C. H., Liu, A., Del Castillo, S., Bowyer, M., Alverson, D., Muniz, G. & Caudell, T. P. (2006). Towards an immersive virtual environment for medical team training. Studies in Health Technology and Informatics, 125, 274-279.

7 Boulos, M. N. K., Hetherington, L. & Wheeler, S. (2007). Second Life: an overview of the potential of 3- D virtual worlds in medical and health education. Health Information & Libraries Journal, 24(4), 233-245.

8 Pausch, R., Crea, T. & Conway, M. (1992). A literature survey for virtual environments – military flight simulator visual systems and simulator sickness. *Presence: Teleoperators and Virtual Environments*, *1*(3), 344-363.

9 Walczak, K., Cellary, W. & White, M. (2006). Virtual museum exhibitions. *Computer*, *39*(3), 93-95.

10 Thomsen, C. W. (1994). *Visionary Architecture: from Babylon to Virtual Reality*. Prestel Verlag, Munich.

11 White, M. et al. (2004). ARCO – an architecture for digitization, management and presentation of virtual exhibitions. Proceedings of Computer Graphics International 2004, 622-625.

12 Hiltz, S. R. (1986). The "virtual classroom": using computer- mediated communication for university teaching. *Journal of Communication*, *36*(2), 95-104.

13 Nussbaum, M., Susaeta, H., Jimenez, F., Gajardo, I., Andreu, J. J., Villalta, M. & Nordlinger, J. (2009). Classroom multiplayer presential games. *Proceedings of the IEEE International Workshop on Technology for Education*, 32-35.

14 Villalta, M., Gajardo, I., Nussbaum, M., Andreu, J. J., Echeverría, A. & Plass, J. L. (2011). Design guidelines for classroom multiplayer presential games (CMPG). *Computers & Education*, *57*(3), 2039-2053.

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

16 Salomons, O. W., van Houten, F. J. & Kals, H. J. J. (1993). Review of research in feature-based design. *Journal of Manufacturing Systems*, *12*(2), 113-132.

17 Smith, B. (Ed.) (1979). Proceedings of the 1979 CAM-I International Spring Seminar, New Orleans, Louisiana, USA, April 10-12, 1979.

18 Wingård, L. (1991). Introducing form features in product models: a step towards CAD/CAM with engineering terminology, *Doctoral dissertation*, Royal Institute of Technology, Stockholm, Sweden.

19 Pratt, M. J. (2001). Introduction to ISO 10303 – the STEP standard for product data exchange. *Journal of Computing and Information Science in Engineering*, *1*(1), 102-103.

20 Hower, W. & Graf, W. H. (1996). A bibliographical survey of constraint-based approaches to CAD, graphics, layout, visualization and related topics. *Knowledge-Based Systems*, *9*(7), 449-464.

21 Houshmand, M. & Valilai, O. F. (2010). Collaborative information system architecture for CAD/CAM in new product development based on STEP Standard. *Proceedings of the World Congress on Engineering and Computer Science*, *2*.

22 Zhang, S., Shen, W. & Ghenniwa, H. (2004). A review of Internet-based product information sharing and visualization. *Computers in Industry*, *54*(1), 1-15.

23 Aziz, E-S., Chang, Y., Tumkor, S., Esche, S., & Chassapis, C., Adopting computer game technology to support engineering laboratories, *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'10*, Vancouver, British Columbia, Canada, November 12-18, 2010.

24 Rachuri, S., Han, Y. H., Feng, S. C., Wang, F., Sriram, R. D., Lyons, K. W. & Roy, U. (2003). Object-oriented representation of electro-mechanical assemblies using UML. *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, 228-234.

25 Aziz, E-S, Corter, J., Chang, Y., Esche, S., & Chassapis, C., Evaluation of the learning effectiveness of game-based and hands-on gear train laboratories, *Frontier in Education Conference*, Seattle, Washington, USA, October 3-6, 2012