A Rapid Development Method of Virtual Assembly Experiments Based on 3D Game Engine

Wenfeng Hu[1, a], Xin Zhang[2,b]

[1]School of Computer Science, Communication University of China, Beijing, China

[2]School of Computer Science, Communication University of China, Beijing, China

[a]huwf@cuc.edu.cn, [b]yixuezx_1988@sina.com

Keywords: Virtual Assembly; Game Engine; Unity3D.

Abstract. Virtual assembly is a promising application in education, design and manufacturing. Traditional virtual assembly applications are based on virtual reality technology. Based on 3D game engine, this paper proposes a new method of building virtual assembly applications, which is more efficient than traditional technology. This method fully takes advantage of 3D game engine, and applies software component technique to construct a reusable component library. Based on this component library, one can rapidly develop a virtual assembly experiment to reveal the structure of a complex object, which is especially suitable for educational purpose.

Introduction

With the development of game industry and the popularity of computer-aided instruction, educational game as a new educational media is attracting more and more widespread interest. As the product of education and games, educational game will improve the traditional teaching mode, and integrate more vivid teaching form and richer teaching contents into it, to realize the educational thoughts of "learning with entertainment". As M. Prensky recounts in his 2001 book [1], "The true 21st century learning revolution is that, learning through playing and playing to learn."

In abroad, the educational game research has an early start. In the 1980s, some scholars proposed that the concept of "educational game" in the course of researching the video game in America. In educational game design and development theory, foreign scholars have done a lot of work, and proposed some development models and methods. K. Kiili's educational theory and experiential game model, and A. Amoryl and R. Seagram's expedient way of designing educational games "GAP" [2] are more famous. In practice of educational game, the most representative is the Game-to-Teach (GTT) Project, which is a partnership between MIT and Microsoft to develop conceptual prototypes for the next generation of interactive educational entertainment. They have developed a dozen conceptual frameworks of games for math, science and engineering education. At home, the educational game research is still at the stage of exploration, which is manifested as follows: the fundamental research of educational game and the practice of educational game software [2].

As an effective teaching method, experiment is an important part of teaching, which has a great significance for us to know the things, consolidate our learning, improve our ability to analyze and solve the problems, cultivate our handling capability and develop a spirit of innovation. Assembly experiments account for a certain percentage in many experiment forms, especially in study of some system structures.

Traditional assembly experiments are usually done with the help of the material objects in real situations [3]. Before experiment, we also need to be trained by experienced technicians. The traditional assembly is indeed intuitive, but we have to spend a lot of time to be trained and worry about the equipment spoilage, the security problems and other limiting factors, such as the manpower, the space, the environment and equipment.

Virtual assembly is based on computer technology, graphic rendering technique, animation technology, multimedia technology, network technique, virtual reality technology, etc [4]. It creates a vividly virtual environment based on the real environment, which may be not realized in reality. The virtual assembly will break the restrictions of equipment, time and place. It is convenient in large-scale and long term assembly experiment, and now, it is very important research topic of education and virtual manufacture [5].

Usually, virtual assembly applications is based on virtual reality technology that have many disadvantages, such as low developing efficiency, bad sense of reality, poor interactivity and insufficient supporting for physical simulation[6]. By comparison, 3D game engine is a more promising platform to develop virtual assembly applications, which provides more technical support for virtual assembly, such as a great graphics rendering effect, a strong physical engine and a powerful editor, etc. Furthermore, it supports real-time rendering and editing, so that developers can modify game scenes at any time.

In this paper, we propose a new technique to develop virtual assembly applications based on 3D game engine. This kind of applications can reveal the composition structure of a complex object by demonstrating its assembly and disassembly process precisely and vividly. This technique fully takes advantage of 3D game engine, and involves a set of methods and reusable software components, which solve the key problems of virtual assembly, such as the expression of assembly relationship, the assembly animation and the camera control, etc. Based on this technique, one can develop a virtual assembly experiment rapidly and flexibly.

The rest of the paper is organized as follows: Section2 introduces Unity3D game engine. Section3 describes the design of the virtual assembly component library based on Unity3D. Section4 presents an example about the application of the virtual assembly component library. And finally in section5, we summarize the paper and make some conclusions.

Introduction to Unity3D Game Engine

This method is implemented based on Unity3D game engine. Game engine is a game development platform, and Unity3D is a popular one, which also supports the development of virtual assembly experiments. The specific features of Unity3D include an integrated editor, cross-platform release, terrain editing, shading, scripts, networking, physics, version control and other features.

Scripting in Unity3D Game Engine. In Unity3D game engine, all objects in a game scene are instances of *GameObject*. A gameObject can be attached to many components which specify various properties. The components attached to a gameObject are inherited from *Component* class. The scripts written by programmers are inherited from *MonoBehavior* class which is a special subclass of *Component*. Fig.1 describes the relationship of *GameObject*, *Component* and *MonoBehavior* in Unity3D game engine in the UML language [7].

In Unity3D game engine, *Camera* is a special *GameObject*. If you want to control the camera, you can write scripts (*MonoBehavior*), and attach them to the camera as components.

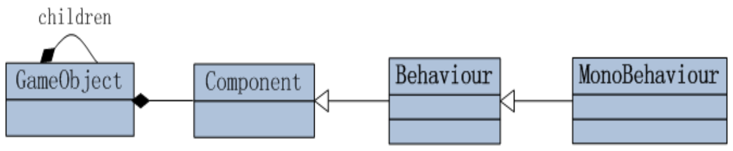
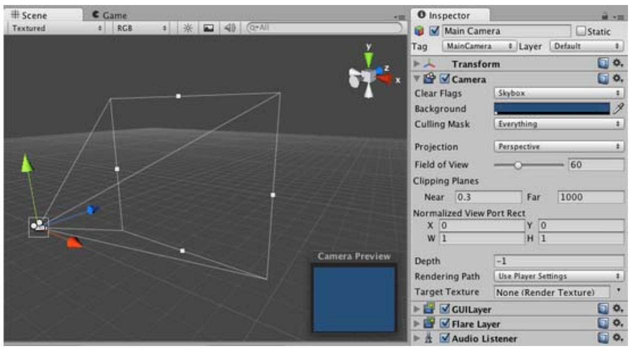


Figure.1The relationship of *GameObject*, *Component* and *MonoBehavior* in Unity3D Game Engine.

Technical Supports for Virtual Assembly from Unity3D

Virtual Camera. All game engines provide supports for virtual camera to some degree, and so does Unity3D. Most importantly, Unity scripts can be attached to or removed from a camera at run time, which allows us to control camera more flexibly. Unity camera has a variety of properties, which can be assigned at design time through game editor and be changed at run time by scripts. With the camera, we can observe assembly parts at the suitable position and angle. Fig.2 shows the camera's properties in Unity3D game editor.



Main Camera
Static
Tag MainCamera Layer Default
Transform
Camera
Clear Flags Skybox
Background
Culling Mask Everything
Projection Perspective
Field of View 60
Clipping Planes
Near 0.3 Far 1000
Normalized View Port Rect
X 0 Y 0
W 1 H 1
Depth -1
Rendering Path Use Player Settings
Target Texture None (Render Texture)
GUILayer
Flare Layer
Audio Listener

Figure 2 The property interface of camera in Unity3D Game Engine.

Collision Detection. Collision detection is a key technology of virtual assembly. In the assembly process, we must determine collisions between gameObjects and respond to collisions rapidly. Otherwise one gameObject will pass through or lap over the other, and it will not be accord with objective law.

In traditional virtual assembly, we usually use algorithms to solve collision detection, such as bounding box method, space partition method and image space algorithm. But in Unity3D, we can add a *Collider* to the gameObject directly, without using complex algorithms. Also, *Collider* is one of *Components*. It not only decreases the developing difficulty, but it also shortens the working hours.

ITween. *ITween* is a simple, powerful and easy to use animation plug-in for Unity with a rich set of animations. It is a single C# file that can be used with any of the programming languages that Unity supports, as well as all version of Unity [8].

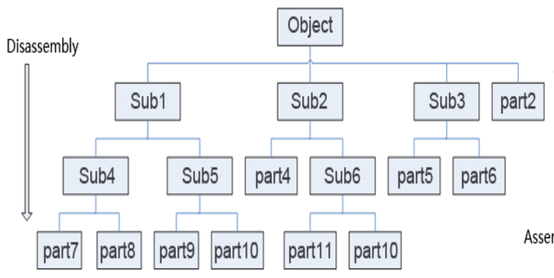
Compare with the animation editor of Unity3D, the best advantage of *iTween* is "one step to the position", that is to say, we can use a simple function to complete the assembly animation. *ITween* is easy to use and save game resources.

The Design of Virtual Assembly Component Library Based on Unity3D

The Assembly-Disassembly Control Script (ADCS)

Assembly Relationships. Complex objects include lots of parts, and they have effects on each other. The assembling condition of some parts may determine whether other parts should be assembled directly. In this system, we design the assembly sequence in advance.

Generally speaking, there are two ways to assemble parts. One is "Whole-Subassembly-Part", the other is "Whole-Part". The main idea of the former is that the object can be disassembled into several subassemblies and parts, and each subassembly can also be disassembled into parts. And the assembly is on the contrary. The latter no longer has subassemblies, and it will handle parts in a unified way. By definition of *preparts* for each part, we indicate the assembly sequence. *Preparts* is a set of parts, only each of which is assembled or disassembled, the related part will be assembled or disassembled. In this system, we use the latter. Fig.4 and Fig.5 show two assembly methods with tree structure.



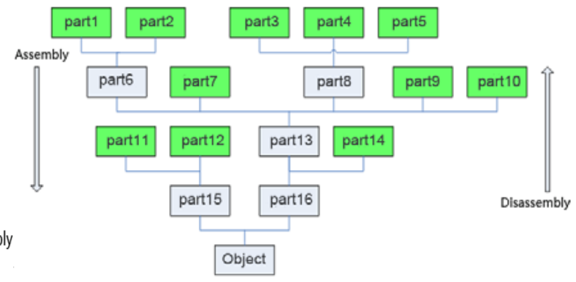


Figure 3 "Whole-Subassembly-Part". Figure 4 "Whole- Part".

In Fig.5, we mark leaf parts with green color, which do not have *preparts* and can be disassembled directly without sequence. In this figure, we take the disassembly as an example: "part1" and "part2" are *preparts* of "part6", and this means that "part6" can be disassembled only after "part1" and "part2" are all disassembled. On the contrary, "part15" is a *prepart* of "part11" and "part12" in the assembly process, therefore, "part15" must be assembled before "part11" and "part12".

Assembly-disassembly Animation. Generally, the virtual assembly process is realized by complex algorithms, such as *Visible Map*, *Transformation Line*, and *Configuration Space Approach*. But in this system, we use the third-party plug-in *iTween*, to design the animation in advance. We can use a simple function of *iTween* to complete the assembly animation, and it will decrease developing difficulty and save game resources.

With *iTween*, we defines motion path of assembly and disassembly for each part at design time, including the original position, the final position and some key positions. Also, we design the original angle and the final angle of each part in advance. Then *iTween* will calculate a smooth curve according to the key points. When the part is assembled or disassembled, it will move or rotate smoothly from the original point to the final point as we defined at design time.

Track Display. Most virtual assembly simulations only plan the assembly path. In fact, players can't keep up if the part moves so fast. As a result, we implement the track display to show the motion path of parts in the assembly process by particle system of Unity3D. By adding a Trail Render component to parts, we can define parameters of track display, such as *Materials*, *Time*, *StartWidth*, *EndWidth* and *Colors*.

The Camera Control Script (CCS)

Virtual camera plays a key role in the assembly process [7]. In this system, we use mouse and keyboard to control virtual camera, and define a set of input operations corresponding to camera motions. For example, you can select a part on the screen by moving the mouse pointer to that part and then clicking left key of mouse.

In the assembly process, the most important thing for camera is to follow the best observation point and angle. At runtime, the point and angle of virtual camera are up to its current position and target object. When some part is selected, the camera will move to the best observation point from current position smoothly with looking at it. Certainly, you can also change the observation point and angle whenever.

In addition, there are some auxiliary components, such as start menu, state transition component, progress component and part property component. Using these components, we can switch back and forth between assembly and disassembly at run time, know the current assembly-disassembly progress and properties of parts. Fig.6 describes this system structure.

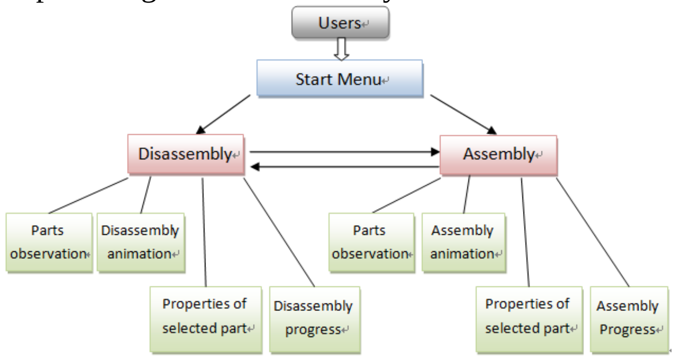


Figure 5 Virtual assembly system's structure diagram.

The Configuration Steps of Rapid Development of Virtual Assembly Systems

With above components, developing a virtual assembly system typically includes the following steps:

(1) Prepare all kinds of auxiliary points: Add different kinds of points to the game scene, such as the best observation point (*PositionPoint*), the best observation angle (*LookPoint*), the motion path point (*MovePoint*) for each part, the real-time observation point (*LookTarget*) for camera and so on.

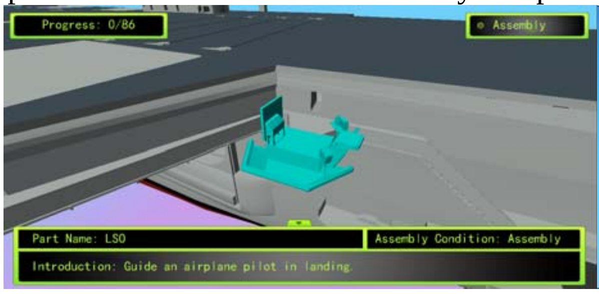
(2) Prepare parts: Attach the ADCS to each part, and configure parts individually. Also, we need to attach a Trail Renderer component to each part.

(3) Prepare the virtual camera: Attach the CCS to the camera object. For configuring conveniently and efficiently, we attach the auxiliary components to the camera object also.

In above process, step (1) will cost much time because of a great deal of points. In order to improve the configuration efficiency, we encapsulate the same kind of points into a "Prefab" (a mechanism of Unity3D), which can be stored, shared and reused easily. We only need to adjust the position of points after adding a prefab to the game scene without adding points by hand.

The application of this virtual assembly component library

Recently, we developed a virtual assembly experiment using this method named "Chinese Aircraft Carrier Varyag". In this experiment, we use the camera control component (CCS) to select a part and observe the part from the best position (Fig.9). Then we use the assembly-disassembly control component (ADCS) to simulate the assembly process (Fig.10). All the functions in this game are completed based on the virtual assembly component library.

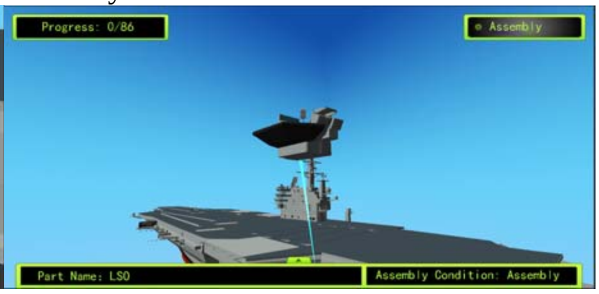

Progress: 0/86

Part Name: LSO

Assembly Condition: Assembly

Introduction: Guide an airplane pilot in landing.

Figure 6 Select a part.


Progress: 0/86

Part Name: LSO

Assembly Condition: Assembly

Figure 7 Assembly animation and track display.

Summary

In education, displaying an object's structure to students by assembling and disassembling is a familiar task. This paper proposes a new technique to build virtual assembly experiment. Using this technique is efficient, because it is based on 3D game engine and all software scripts involved in it are made into reusable components. This technique is also ready to use, because all necessary processes to build a virtual assembly experiment are explained in details.

References

[1] M. Prensky: *Digital Game-Based Learning*(Paragon House Publishers, New York 2001).

[2] H.Y. Yang: The Recent Development of Educational Game, Modern Educational Technology, Vol.19, p.26-31. (2009)

[3] J. Zhang and Z.J. Zhai: Research and Application of Virtual Assembly Technology, Aeronautical Manufacturing Technology, Vol.1, p.70-73. (2009)

[4] L. Gao and H.J. Dong: Recent Study Progresses and Trends of Virtual Assembly Technology, Inner Mongolia Science Technology & Economy, iss.20. (2009)

[5] L.L. Yao, X.Q. Li and C.J. Yao: Research and Implementation of Virtual Assembly Training System, IT in Medicine & Education, Vol.1, p.641-647. (2009)

[6] F. Zhang: Research and Implementation of Mechanical Power Simulation Technology Based on Game Engine, Software Guide, Vol.10, p.83-86. (2011)

[7] W.F. Hu and X. Zhang: A Semiautomatic Control Technique for Machinima Virtual Camera, International Conference on Computer Science and Electronics Engineering, p.112-115. (2012)

[8] Information on http://itween.pixelplacement.com/index.php