An Improved Approach for Mechanics Simulation Based on Game Engine

Wenfeng Hu[1, a], Zhouqing Qu[2,b] and Xiaoyuan Zhang[3,c]

[1]Department of Computer Science, Communication University of China, China

[2]Department of Computer Science, Communication University of China, China

[3]Department of Computer Science, Nankai University Binhai College, China

[a]huwf@cuc.edu.cn, [b]frankieqing@163.com, [c]zxy_nkbh@163.com

Keywords: mechanics simulation, physics engine, game engine, component library, assembled mechanical devices

Abstract. Machinery dynamics is an important branch of physics based on experiments which usually are inconvenient and expensive in reality. Computer simulation is a reasonable substitute for real experiments, but traditional mechanics simulation approaches also have many disadvantages, such as low interactivity, low flexibility, low reusability, high development difficulty and etc. With the introduction of game engine technology, especially the physics engine technology, mechanics simulation based on game engine achieves good effects. This paper proposes a method to build a reusable and interoperable component library for mechanics simulation based on game engine. Users can rapidly and flexibly set up various mechanics experiments at run-time by reusing these components as building-blocks. Due to the high interoperability and interactivity, these building-blocks can be easily composed into complicated machines at run-time.

Introduction

As an important branch of classical physics, the machinery dynamics is a subject which is valuable in study. Like most branches of physics, the machinery dynamics is based on experiments. Through mechanics experiments, researchers can understand the mechanism of the mechanical devices more thoroughly. However, real apparatus of experiment have some disadvantages. For example, real apparatus may be expensive and easily damaged. Moreover, a set of experimental apparatus may occupy a large volume of space and only serve a limited number of researchers simultaneously.

With the development of computer science, people found that the integration of computer simulation technology and virtual reality technology may offer a suitable substitute for real mechanical experiment. Generally, traditional method to build a virtual mechanical experiment involves the following steps: First, developers make 3D models for all apparatus involved in an experiment with some 3D modeling software (3DMAX, AutoCAD, Pro/E, etc). Second, the 3D models should be imported into some 3D graphic real-time rendering platforms (such as VRML, Cult3D, etc) which make a visual simulation for mechanical experiment. Furthermore, almost all 3D graphic real-time rendering platforms provide interfaces for developers to write scripts to define the behaviors of the virtual objects in VR scenes [1].However, the traditional method has some disadvantages as follows.

- The high development difficulty. Developers need to write scripts to determine the instantaneous state of the mechanical devices at any time, including the geometric position, the direction and the shape. That decides the high difficulty of the simulation.
- The low interoperability. In traditional simulation, it is totally the script that keeps the mechanical models running, which makes the running process rough. In that way users can't learn any physical significance from the simulation.
- The low reusability. The running effects of the objects are controlled by code. So according to different demands developers need to do different coding, which can't show the reusability.

Published by Atlantis Press, Paris, France.

- The low interactivity. The traditional simulation can't support more interactivity for users. At run-time, users can only receive the visual simulation effects from the system passively. They can't interact with any object.

With the development of the game engine technology, especially the physics engine technology, we put forward a subject that whether the game engine can be used to perform mechanics simulation. Experiments prove that the answer is yes. The physics engine provides a set of basic functions for physical simulation. Developers just need to implement the functions in a proper way. In other words, the physics engine can make the objects act under the control of the physics laws which is done by programmers in traditional techniques without physics engine. Then, it becomes easier to develop some applications based on physics laws, such as mechanics simulation. However, mechanics simulation applications based on game engine are all once-off applications. It means that the objects that have been simulated in one application can't be reused in another one. That's exactly what we need to improve.

The Italian physicist Galileo had ever proposed a theory that each of the complex machines in nature is composed of several classical simple machines. Inspired by this theory, in this paper we build a reusable and interoperable component library which comprises various forces, some classical simple machines and the interactive module. At run-time users can assemble the components in the library into a whole complex mechanical device through the interactive module without any extra programming. Then the device will run under the control of the interactions between the interfaces of the components. That provides a more convenient way to simulate mechanical devices for the developers [2]. Through experiment it has been proven that the new method can improve the development efficient and increase the interoperability, the reusability and the interactivity of the simulation.

Introduction to unity3d game engine

Among so many game engines, in this paper we choose unity3d as our developing tool. For the topic of this paper—mechanics simulation, the physics engine is exactly what we concern. So here it is worthwhile to discuss the physics engine of unity3d in detail. In unity3d, the hierarchy of the physics engine can be described as Fig. 1. Here we introduce some core components which may be frequently used in the simulation.

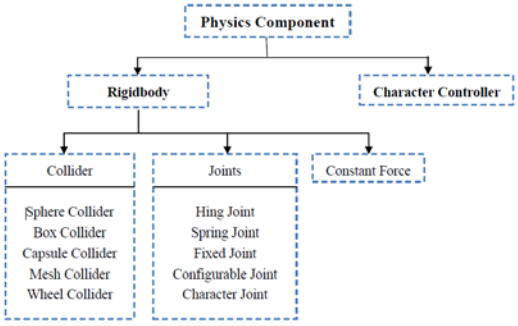

Figure1. The hierarchy of the physics engine

Rigid body: This component is the foundation of other components. It enables the game object to be under the control of physics. It can receive forces and torques to make the objects move in a realistic way.

Collider: Whereas the rigid body component allows objects to be controlled by physics, collider allows objects to collide with each other. This component enables the object to receive the collision signals.

Constant Force: It can be used to add constant forces to an object. When an object is influenced by external forces, it is necessary to add this component to it to simulate the situation.

Joint: This is a component that can simulate the conjunction relation between adjacent parts of the models. Due to difference in types of the connection, it can be divided into fixed joint, hinge joint, spring joint, configurable joint and character joint.

Published by Atlantis Press, Paris, France.

Unlike the traditional technique, mechanics simulation based on unity3d game engine has many advantages:

- The low complexity. The game engine provides a framework for the game. The developers only need to add related modules to the framework. Then the framework can control the different kinds of data to show their different functions. In other words, programmers assign the complex task to the physics engine instead of doing that by themselves. That simplifies the work of developing a mechanics simulation system greatly.
- The high interoperability. With the help of the engine, designers can build all parts of the machine easily. All the things they need to do is to implement interfaces for each part of the machine. Then the parts can interact with each other through interfaces under the influence of the physics engine. As a result of that, the mechanical devices run as well as expected. That means that it is the interaction between the parts, not the code that controls the mechanical devices to work, which greatly shows the high interoperability [3].
- The high reusability. There is a theory on which this paper is based that all complex machines are combinations of the simple machines. In this paper we design a component library which consists of all kinds of simple machines. When making mechanical simulation, developers can directly reuse the components in the library.
- The high interactivity. The traditional techniques only have limited interactive function which can be seen only at design-time. Unlike the traditional techniques, the system based on the game engine can provide more interactive forms at run time through the interactive module.

The reusable and interoperable component library for mechanics simulation based on Unity3D game engine

The main purpose of this paper is to build a component library for mechanics simulation. The components in the library can be reused at run-time. Elements of the library fall into three categories: the first one comprises sources of various forces, such as gravity, spring, friction and etc; the second comprises all classical simple machines, such as Lever, Wheel and axle, Pulley, Inclined plane, Wedge, Screw and etc; the last section comprises the interactive module through which users can interact with the mechanical devices at run-time. All of the elements have a unified interface through which it can interact with other elements. In essence, the interfaces are equivalent to the forces in real world, while implementing the interfaces is equivalent to simulating the forces in real world. At design-time, developers just need to get the components out of the library and configure the interactive module for the users. Then at run-time users can assemble the components into a whole model through the interactive module. As a result, the whole device will run under the control of the physics laws. That fully shows the reusability, the interoperability and the interactivity of the new method.

The simulation of the forces.

Gravity. Unity3d provides a direct way to simulate gravity for users. The Rigid body component contains a Use Gravity attribute. Developers only need to set the value of the attribute to true.

Friction. There are several methods to simulate friction. Firstly, developers can simulate the contact surface with physics materials. If the surface is added physics materials, it can exert frictions to the moving objects on it. However, by this means friction only can be simulated roughly. If an accurate simulation is required, that method is not applicable. In that case developers can add a Constant Force component to objects and set the magnitude and the direction of the friction in the component [4].

Air resistance. To simulate the air resistance, developers need to set the values of the drag and the angular drag in the rigid body component. The value of the drag decides how much air resistance affects the object when moving from forces. 0 means no air resistance, and infinity makes the object stop moving immediately.

Elasticity. Unity3d provides an easy solution to simulate the elasticity produced by collision. Developers can add a collider component to objects that may collide with others. Then the object will

own the function of collision detection. That means that it can apply elastic forces to the object which is in contact with it.

The classical simple machines.

Lever. Here we take the balance as an example of the lever. The balance is a kind of lever which can neither save the labor nor waste the labor. It can be used to weigh something. The objects which need to be weighed can be put in the left plate, while the weights can be put in the right plate. Then according to the mechanics laws we can register the weight of the objects. Fig. 2 is the final simulation drawing.

Wheel and Axle. An example of the wheel and axel machine is the windlass which can be used in old-fashioned wells to hoist up the water bucket. A windlass contains two parts: the rotary axis and the rotary beam. The beam rotates under the action of external forces, which simultaneously drives the axle to rotate around the same axis. Then the rope around the axle will move upwards. As a result the objects at the other end of the rope will be hoisted up.

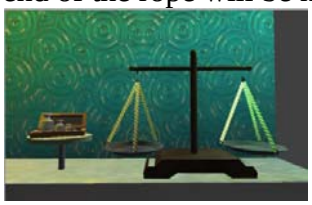

Figure 2. Balance

Figure 3. Windlass

Inclined plane. With a moving belt on it, the inclined plane can help people to transfer objects. The physics principle is as follows. When put on the moving belt, the object has a downward tendency relative to the inclined plane. Then the belt applies an upward friction to the object. It is the friction that makes the object move with the belt together.

Gears. Several different gears can be combined to be used as mechanical driving devices. The physics principle is: When one of the gears rotates under the action of the external forces, by the frictions the gears meshing with it will rotate together with it. The gear that rotates first we call drive gear. The gears driven by the drive gear we call driven gear.

Figure 4. Inclined plane

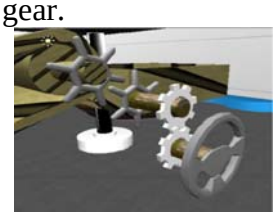

Figure 5. Gears

The interactive module. In order for users to assemble the components at run-time, it is necessary to design an interactive module for our library. Through the interactive module users can interact with the mechanical parts to assemble them into a complex machine at run-time. If the assembly process is reasonable, the machine can run under the control of the interactions between the interfaces of the simple devices. The interactive module can be divided into three sections which exactly represent the three technical points [5].

Operate the components at run-time. Unity3d has its interactive module which built up a bridge between users and the system. The module can receive input signals from different devices and process the signals according to some rules. In other words, different input forms represent different meanings. The only thing developers should do is to tell the system the rules. Then the system knows how to process the input and what to response to the users. In that way users can interact with the models.

Therefore, here we can write script in the function On Mouse Drag to tell the system when the users operate the mouse the parts of the machine can be operated according to different input type. And the script component should be added to every part in the whole machine. Then the part can be operated under the control of the mouse.

Align one component with another at run-time. We've known that the simulation is based on the interoperability between components. However, how to determine the time point one component can interact with another one is still a problem. So in the interactive module we provide a method to

align one component with another. Then the two components can interact with each other. Here we choose the collision detection technique to help us. To be specific, we should add a collider box to each of the components. At run-time when one part enters the box of another part, the part can detect that. We can assume that the two components are aligned with each other.

Implement the interface dynamically according to different machine combinations. When the engine detects that one component tries to be align with another one, it will implement the unified interface in different ways according to the different types of the two components. In other words, at run-time the engine can dynamically add different physics components to different simple machine combinations, which is under the control of our script. Then it can be considered that the engine establishes a connection between the two devices. Finally the two devices can interact with each other. As a result, the whole complex machine can run as we expect. Here we give some examples which can explain how to implement the interface [6].

Lever and lever: Generally, there is a connection relationship between two adjacent levers. At run-time, we should dynamically implement the interface to simulate the relationship. Specifically, the hinge joint component should be added to the two levers after one lever enters the collider of another one. Then the two levers can interact with each other as we expect.

Gear and gear: One of the gears is the driving gear and the other is the driven gear. When the driving gear rotates, the driven gear can rotate following it. That's exactly what we need to simulate at run-time. Here we should dynamically add the configurable joint component to the driving gear to make it rotate. Then the driven gear can be driven to rotate under the control of the collision detection script.

Lever and gear: The lever can transmit forces to the gear. Then the gear can rotate around the axis. First, we need to add a fixed joint component to both of the lever and gear to make sure that they are connected to each other. In addition, a configurable joint component should be added to the gear. Then it can be ensured that the gear can rotate rather than move.

Pulley and inclined plane: An inclined plane with a pulley can be used to transfer objects. In this mechanical combination, both of the two parts should have a fixed joint component. Besides, the inclined plane should be dynamically added a script which can judge whether an object touches it. If there is an object on it, it can add a force to the object. Then the object can be transferred from one edge to the other.

Similarly, we can simulate other mechanical combinations to enrich the content of the library. Finally, users can reuse the library to assemble into complex mechanical devices at run-time. That's what we want to achieve.

The simulation of complex mechanical devices assembled by classical simple machines at run-time

The Italian physicist Galileo had ever proposed a theory that each of the complex machines in nature is composed of several classical simple machines. So we can reuse the library to assemble all kinds of mechanical models. It is very easy to reuse the library. Before the run-time, users should prepare the models and configure the interactive module according to the device's machine combinations. And then we put the models in the corresponding positions in the engine. At run-time, users can interact with the elements through the interactive modules. Then the interfaces of the elements can be dynamically implemented. Through the interfaces the elements can receive forces from others and apply forces to others. After that, all elements in the environment can work together in harmony. That means that the whole device model can run in control of the physics engine [7].

Here we take an example to prove that the method is available. We simulate a complex conveying device, as shown in Fig. 6. The device can convey objects from one container in one end to another container in another end. In this experiment, we reuse several elements in the library, such as inclined plane, lever, gears and pulley. At run-time users can drag the parts to assemble them into a complex device like Fig. 6. Then if the assembly work is exactly correct, the parts can interact with each other through their interfaces. Finally, the whole device will run.

Published by Atlantis Press, Paris, France.

Figure 6. The complex assembled conveying device

Experiments have proven that the introduction of this library can improve the reusability, the interactivity and the interoperability of mechanics simulation greatly.

Summary

Considering the defects of the traditional mechanics simulation, this paper proposes a new approach for mechanics simulation: Use the 3D game engine to be the basis of the simulation. In addition, this paper also builds a reusable and interoperable component library for mechanics simulation, which provides a convenient way for users to simulate the various machine devices at run-time. The research has reached the expected effect. It has been proven that the mechanics simulation based on 3D engine not only is available but also has more interoperability, interactivity and reusability than the traditional techniques. Besides, the introduction of the reusable and interoperable run-time component library makes the development process become more efficient.

References

[1] Z. G. Pan , A. D. Cheok , H.W. Yang , J. J. Zhu and J. Y. Shi, Virtual reality and mixed reality for virtual learning environments, Computers & Graphics, 30(1) (2006)

[2] P. Greenwood, J. Sago, S. Richmond and V. Chau, “Using game engine technology to create real-time interactive environments to assist in planning and visual assessment for infrastructure” 18th World IMACS / MODSIM Congress, Australia, July (2009).

[3] V. Tam, Z. X. Liao, C.H. Leung, L. Yeung and C.M.Kwan, An Interactive Simulation Game to Enhance Learners’ Experience on Ubiquitous Computing Devices.

[4] H. Y. Du, J. Zheng, J. Wang and X. Y. Tian, A Control Mechanism of The Physical Simulation Experiment based on Game Engine ,Computation Intelligence and Software Engineering, International Conference. pp. 1-3. (2009)

[5] L. Bishop, D. Eberly and T. Whitted, Designing a Pc GameEngine Computer Graphics and Applications，IEEE, pp. 46-53. (1998)

[6] J. Jacobson and M. Lewis, Game engine virtual reality with CaveUT, Computer, 38(4) (2005)

[7] J. Preece, Y. Rogers and H. Sharp, Interaction design: Beyond human-computer interaction, New York, (2002)

Published by Atlantis Press, Paris, France.