Automatic Real Time Simulation through Embedding Physical Attributes

http://dx.doi.org/10.3991/ijoe.v9i6.3015

Tan Xiaohui [1,2], Li Shanshan [1] and Zhou Mingquan [1]
[1] Beijing Normal University, Beijing, China
[2] Capital Normal University, Beijing, China

***Abstract*—An automatic real time physical simulation method is presented through embedding physical attributes into collections of 3D models in Virtual Rapid Scene Building System called VR Scene Studio. In the physical simulation, dynamic equations of the objects with physical attributes are established by Lagrange method. The improved impulse-based paradigm is adopted to gain interactive simulating speed. To improve the performance, optimizing methods are investigated. The real time and physically plausible result can be achieved by the combined the above two methods. The experiment results show that the method is valid and practical.**

***Index Terms*—Physical attributes, real time simulation, scene building system, virtual reality.**

I. INTRODUCTION

A scene rapidly building system, including model generation, textures, and animation and so on, plays an increasingly important role in many applications, from computer games and 3D virtual wandering to urban planning. A scene building system can be used to build interactive and more realistic environment with low cost and less time.

Realism with physical attributes such as gravity, friction, collision, etc. is an important for a scene building system. Without physical attributes, models in the Virtual Scene will not fall down from high, and two objects will interpenetrate each other. Models in the scene should show the appropriate behavior according to their physical attributes. For instance, objects have different velocity depending on their height and mass, and two or more objects probably collide with each other. In a virtual scene, the ability is necessary to accurately predict the motions and deformation of objects with/without external force. Because contact and frictional behavior, dynamic models should be used to gain a realistic virtual scene [1].

As such importance, virtual reality tools often make use of physical simulation through certain physics engines. The most popular open source physics engines like Bullet [2] and Open Dynamics Engine [3] provide such functions. Academic and commercial developers also create their applications based on commercial physics engine Nvidia PhysX or Havok. But many developers do not have the know-how and resources to develop general purpose physics engines, as witnessed by the recent growth in the physics middle-ware industry.

Literature and academic interests in supporting rapid scene building are growing. Liu et al. [4] develop a Virtual Reality platform, designing server and client applications, including a rendering engine based on OGRE, and finally they exhibit an interesting application. But their physical supported attributes are not mentioned. Braun et al. [5] presents a platform for simulating virtual environments where users can communicate and interact with each other using avatars with facial expressions and body motions. Some physical effects have been implemented to increase the realism of the simulation. In most VR system, Objects have only geometric representation, and no physical attributes are attached. As for rigid body simulation, O'Reilly Media, Inc. and Kaufmann provided a "big picture" view of a practical implementation [6, 7]. Most of the literature of physical simulation is not based on 3D objects database [8, 9].

VR Scene Studio is a 3D rapid scene building system developed by the team of Institute of Virtual Reality and Visualization Technology in Beijing Normal University[]. The System is based on the Massive 3D Database and 3D scene can be built by it with less time than traditional tools. But currently there are no physical attributes in the scene.

Our physical simulation method will solve such problem. It is different from the common physical engine. It is a pluggable component of the VR Scene Studio. The physical attributes can be embedded to the 3D models for physical simulation. The stability of physics simulation is vital; otherwise the users experience will be ruined. Our algorithm will focus on time stepping simulation methods. Another requirement of dynamic simulation in VR environment is real time. But as the computation is usually large in simulation, accuracy requirement is reduced to achieve real time performance. Due to the visually plausibility, the motion equations can be modified to approximate the simulation.

The primary contribution of this paper is to embedding physical attributes into 3D scene building system as VR Scene Studio to enhancing physical accuracy. It is able to deal with the collision forces with the existing virtual scene building system easily. This paper is organized as follows: Section 2 is the overview of VR Scene Studio and the structure of the physical simulation module's framework. Section 3 is the optimize methods adopted in the physical simulation module. In Section 4 and 5, we present some results along with a brief discussion of specific topics and possibilities for a future work.

II. MECHANISM OF EMBEDDING PHYSICAL ATTRIBUTES

In this section we describe a unified framework for embedding physical attributes into a 3D scene rapidly

building system. We start by the whole architecture of the VR Studio. To embed the physical attributes into the VR Studio, the representation of physical attributes is defined secondly. The dynamic model is the core of the physical module; finally the detail of the dynamic model is introduced.

A. Architecture of the VR Studio

The VR Scene Studio is a 3D rapid scene building system. In this section, we briefly illustrate the components integrated in VR Scene Studio. Figure 1 shows a high-level view of the overall architecture of the scene building environment. Osg is responsible for the rendering of the 3D scene. OsgBullet is the core physical module based on Bullet--an open source physical engine. The module is mainly included of three parts: users can set and get physical attributes through osgbInteraction, osgbCollision is responsible for collision detection and collision resolution. The most important part is osgbDynamics which is the core dynamic simulation class.

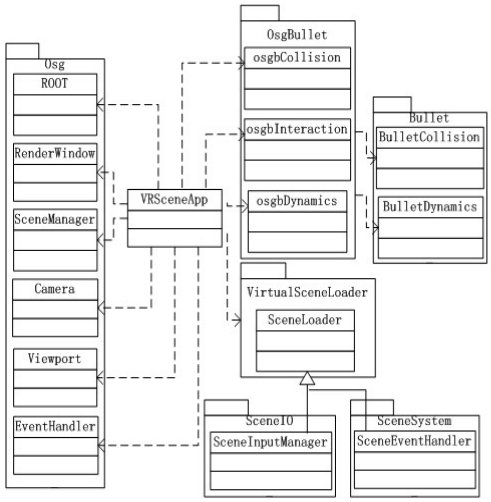

Figure 1. Overall architecture of the scene building environment

An important feature typically provided by the scene building environment is the physical realism. 3D environments are enriched by the physical module which is responsible for setting and getting physical properties, collision detection, collision response, and multi-body dynamic modeling and computation. The structure of the physical module is illustrated by Figure 2.

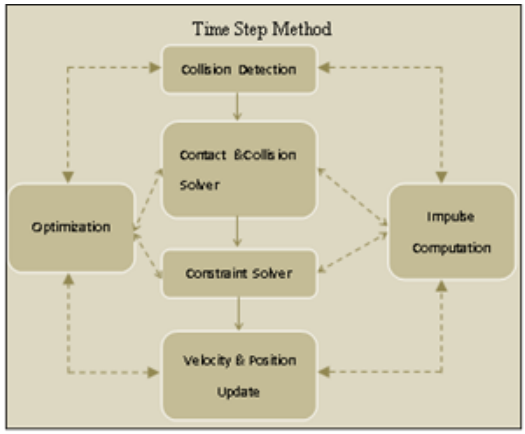

Figure 2. Structure of the physical module

In forward simulation the impulse was computed all through the whole process. The contact/collision solver gets contact/collision information from collision detection module and preprocesses the collected information. Then the constraint solver assurance the constraints to be not violate. The optimization module is used to accelerate the dynamic simulation. One time step later, computed new velocity and position are gained and the status of the system is updated.

Currently, two simulating paradigms are widely adopted to treat the collisions: constraint-based and impulse-based. The latter is adopted in our physical library. Hahn firstly used a series of micro-collisions to prevent penetrations of resting objects [10]. Mirth improved Hahn's method and proposed the impulse-based paradigm [11, 12]. Impulse-based simulation is different from typical interactive dynamic simulation on the way of contacts modeling. Rather than computing explicit constraint forces to be applied at contact points, contact interactions are modeled exclusively through collision impulses applied between bodies. Impulse-based simulation is designed to simulate moderately complex systems at interactive speeds. All interaction between objects is modeled only through contact points. Impulse is applied at these points in the collision response stage.

B. Representation of physical attributes

In this part we describe the representation of physical attributes to embed into the 3D rapid scene building system-VR studio. Based on the Massive 3D Database, every 3D model is treated as an entity in the VR Studio. Each entity is retrieved from database and added into the VR studio. With the support of physical module, the entity is independent functional component and can interact with the virtual environment and other entities. The virtual scene is composed of many entities. The following formal definitions give the representation of physical attributes.

Definition 1： The Virtual Scene is composed of N entities (VSEle).

$$VS=\{ VSEle_i : i \in R \ \}$$

Definition 2：The Virtual Entity is the component of the Virtual Scene. Every Virtual Entity has geometrical and physical attributes. When the Virtual Entity is added into the Virtual Scene, the physical module is responsible to attach appropriate physical attributes with the Virtual Entity.

$$VSEle = \{index, GeomAttr, PhyAttr : index \in R\}$$

Definition 3：Geometrical attributes stand for the information of the geometry.

$$GeomAttr = \{DrawObj_i : i \in R\}$$

Definition 4: The physical attributes describe all the physical features which enable the Virtual Elements interact correctly with other elements in the virtual scene.

$$PhyAttr = \{Attr\_Entry_i : i \in R\}$$

Definition 5: The physical attribute entries including name, category and value of the physical attributes. There exists logical relationship between the virtual entities. Physical module is responsible for maintaining the relationships of the virtual entities to assure the physical feasible features.

$$Attr\_Entry = \{attr\_name, attr\_kind, attr\_value\}$$

$attr\_name$ is the name of the attribute entry.

$attr\_kind$ is the category of the attribute entry.

$attr\_value$ the value of the attribute entry.

Definition 6: Action is the trigger of the entities' transformation. An action is included of five parts as following:

$$Action = \{act\_name, act\_sender, act\_receiver, act\_id, act\_param\}$$

1) Construct the virtual scene entity from the database.

2) Setup the physical attributes.

3) According to the conditions, the physical module responsible for change the physical attributes.

4) The underlying time step method and dynamical model govern the action of the virtual entity.

Figure 3 shows the process of the interaction between the physical module and the database.

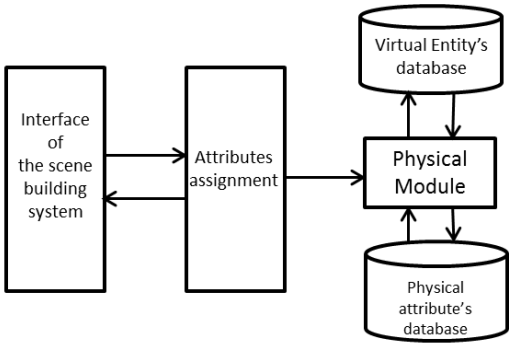

Figure 3. Overall architecture of the scene building environment

C. Mapping physical attributes to collections of 3D shapes

To attach physical attributes to the 3D objects, it should define the mapping between 3D objects and physical attributes. Due to the variety, the 3D model objects are divided into 128 categories, 418 subcategories. As for physical attributes, different types of objects have different physical attributes. According to physical variety, objects are divided into three main kinds: rigid, soft, fluid. There are two types of mapping problems. One is the transformation performed on two categories. The other is the transformation performed on detailed physical attributes and 3D shapes.

The proposed mapping diagram looks like a class diagram. Fig. 4 shows the mapping diagram. The mapping specification shows two levels of mapping which correspond to the former two types of mapping problems.

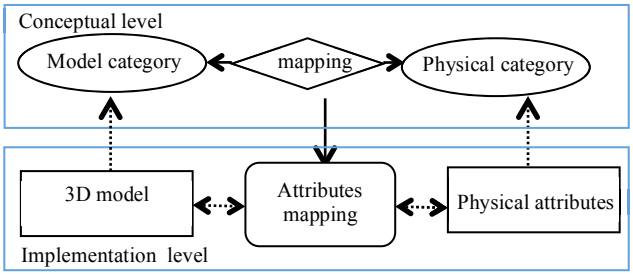

Figure 4. Mapping physical attributes to 3D shapes

Figure 4 shows the mapping procedure which is applied to the mapping between 3D model object and physical attributes. Because the data model and the implementation method are separated, the mapping methodology used for the implementation can be changed easily. For simplicity, we established a mapping table between model category and three physical kinds. As for a specific physical classify, a predefined physical attributes set for next step of mapping. Because variety of physical attributes for specific 3D object even in the same class, the additional artificial modify interface is also provided.

III. DYNAMIC SIMULATION OF OBJECTS WITH PHYSICS ATTRIBUTES

After embedding physical attributes, the physical module is responsible for simulating objects in the scene with physical correctness. Dynamic model is key issue for physical simulation. The dynamic equation is established by Lagrange method and the core Dynamic module is impulse-based paradigm. Section A introduces the detail of the dynamic model. Collision is also important in physical simulation, section B introduces collision response.Due to the length of this paper, the multi-body dynamics [13] is omitted.

A. Dynamic Model

The motion state of a body i can be described by its position of the center of mass $\vec{q}_i(t)$, its orientation $\vec{R}_i(t)$, its linear velocity of the center of mass noted as $\vec{v}_i(t)$,and its angular velocity $\vec{\omega}_i(t)$. So the current state of a body is given by a state vector $\left(\vec{q}_i(t), \vec{R}_i(t), \vec{v}_i(t), \vec{\omega}_i(t)\right)^T$. The simulation carries out and is controlled by the discrete time-steps of size Δt .The goal is to determine the velocity and position of the new time. The position of the object in the 3D scene is described by the global and local coordinate together. In configuration space $SE(3)$, the local coordinate's original point can be described, considering the convex polyhedral property, the position of objects in scene can be described in the coordinate system showed in the Figure 5.

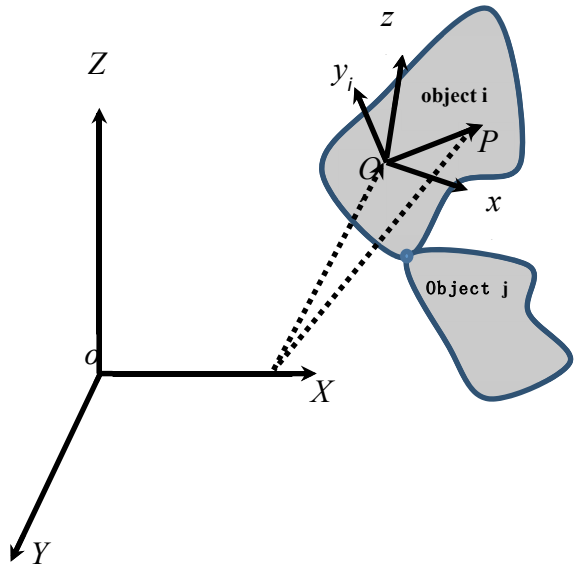

Figure 5. Coordinate System

$OXYZ$ indicates global coordinate, $o_i x_i y_i z_i$ is the local coordinate. Suppose point P is the mass center of the 3D object, then the position of P is denoted as

$$\boldsymbol{q} = \vec{\boldsymbol{r}}_p = \vec{\boldsymbol{r}}_0 + \Theta\vec{\boldsymbol{r}}_{pi} = \vec{\boldsymbol{r}}_0 + \boldsymbol{\Theta}\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad (1)$$

, where $\vec{\mathbf{r}}_0 = \overrightarrow{\mathbf{OO'}}$ $\vec{\mathbf{r}}_0 = \overrightarrow{\mathbf{OO'}}$, $\vec{\mathbf{r}}_{pi}$ and $\vec{\mathbf{r}}_p$ is the position vector in local and global coordinate[14]. $\boldsymbol{\Theta}$ is the transfer matrix of local coordinate related to global one.

Define the generalized coordinate of the system as $q \in R^n$ (n is the number of coordinates). The kinetic energy denoted as $T = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\boldsymbol{M}\dot{\mathbf{q}}$, where $\boldsymbol{M}$ is the mass/inertia matrix of the part. Suppose object j produced contact force as $\sum_{i=1}^{3} F_i I_i$ then the contact force produced by i is, $\sum_{i=1}^{3}(-F_i I_i)$ the Lagrange equation is as equation (2):

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} = Q_i + \sum_{i=1}^{3}\left(\frac{\partial x_j}{\partial \dot{q}} - \frac{\partial x'_j}{\partial \dot{q}}\right)F_i \quad (2)$$

where Q_i is the general force, and does not include the contact force of the contact point. The forces beyond impulse forces are ignored. Take the micro-collision as a dynamic process of period $[t_1, t_2]$, the velocity and tangent impulse are continuously change

with normal impulse. Take the normal impulse as independent variable; the dynamic motion equation can be described as :

$$\begin{cases} \dfrac{dv_t}{d\boldsymbol{P}_3} = K^T\boldsymbol{M}^{-1}K\dfrac{d\boldsymbol{P}_t}{d\boldsymbol{P}_3} + K^T\boldsymbol{M}^{-1}h \\ \dfrac{dv_3}{d\boldsymbol{P}_3} = h^T\boldsymbol{M}^{-1}K\dfrac{d\boldsymbol{P}_t}{d\boldsymbol{P}_3} + h^T\boldsymbol{M}^{-1}h \end{cases} \quad (3)$$

Collision Response

The collision response model is responsible for computing the pair of equal and opposite impulses which applied to the colliding bodies in order to prevent penetrating from each other. Like contact forces, collision impulses are subject to frictional constraints, as well as other constraints governing the energy dissipation during collisions. According to the sliding direction assumption, collision response algorithms can be divided into two categories. The first one assumes sliding direction is constant. This approach is used widely in interactive dynamic simulation [15-17]. A collision impulse changes the relative normal velocity as well as the relative tangential velocities. The direction of the tangential velocity is not constant. The constrained-based method use differential equations that describe the collision process. This class of collision response methods has been studied by many researchers [18-20]. The approach of collision response adopted here is based on the second method. The collision detection library responsible for determine the contact points and the contact normal vectors. Suppose the constraint anchor point are represented by p_a, p_b, and the relative velocity. Suppose there exists K contacts , and i_k, j_k , represent two contact objects. The contact normal vector denote as $\vec{n}_k$, the contact anchor point is $\vec{\boldsymbol{p}}_{\boldsymbol{k}}$, then the relative velocity along contact normal direction is $\vec{\boldsymbol{v}}_{krel} = \vec{\boldsymbol{n}}_k(\vec{\boldsymbol{v}}_{ik} - \vec{\boldsymbol{v}}_{jk})$, where object i_k, j_k at contact point p_k is $\vec{\boldsymbol{v}}_{ik} = \vec{\boldsymbol{V}}_{ik} + \vec{\boldsymbol{\Omega}}_{ik} \times \vec{\boldsymbol{r}}_{k_{ik}}$, $\vec{\boldsymbol{v}}_{jk} = \vec{\boldsymbol{V}}_{jk} + \vec{\boldsymbol{\Omega}}_{jk} \times \vec{\boldsymbol{r}}_{k_{jk}}$. The relative velocity is examined to classify the contact: 1) the relative velocity is greater than zero, this isn not a contact and will not be reported by the collision detection. 2) The relative velocity is equal to zero, a touching contact without penetration. 3) The relative velocity is smaller than zero, this is a penetrating contact.

To prevent objects penetrate with each other, the relative velocity should be satisfied the following condition:

$$\begin{aligned} &\vec{v}_{krel} = \vec{n}_k^T(\vec{V}_{jk} + \vec{\Omega}_{jk} \times \vec{r}_{kj_k}) - \\ &\vec{n}_k^T(\vec{V}_{ik} + \vec{\Omega}_{ik} \times \vec{r}_{ki_k}) >= 0 \end{aligned} \quad (4)$$

When the collision detection model collected the information of collision, then the collision resolve process is called. According to the conservation of momentum Law, the following equations should be satisfied.

$$\begin{aligned} m_i\vec{v}_{i^-} + m_j\vec{v}_{j^-} &= m_i\vec{v}_{i^+} + m_j\vec{v}_{j^+} \\ I_i\vec{\omega}_{i^-} + I_j\vec{\omega}_{j^-} &= I_i\vec{\omega}_{i^+} + I_j\vec{\omega}_{j^+} \end{aligned} \quad (5)$$

In the above equation, the superscript '-' indicates the state before collision, the subscript '+' indicates the state after collision. At the collision time, the impulse act on object i, j is equal but opposite pairs and can be expressed by:

$$\vec{J} = m_i(\vec{v}_{i^+} - \vec{v}_{i^-}) = -m_j(\vec{v}_{j^+} - \vec{v}_{j^-}) \quad (6)$$

Suppose p is the contact point, $\boldsymbol{n}$ denote the normal direction of contact point. $\vec{\boldsymbol{n}}, \overrightarrow{\boldsymbol{n}_x}, \overrightarrow{\boldsymbol{n}_y}$ constitute the standard normal coordinate system. indicates the collision coefficient of friction, the impulse of collision satisfied Coulomb friction law and expressed as inequality of equation (7):

$$(\vec{n}_x \cdot \vec{j})^2 + (\vec{n}_y \cdot \vec{j})^2 \le \mu^2(\vec{n} \cdot \vec{j}) \quad (\vec{n} \cdot \vec{j}) \ge 0 \quad (7)$$

Using one contact point to approximate the contact region is the most practical method .With this method the collision response process will be speeded up. But for most cases, single point represents is a coarse approximation and results in unstable simulation. In our implementation, we adopted the algorithm that defining polyhedral area as the contact region. The drawback of the algorithm is to require more computation time. In one step time, the contact points are calculated. In next one, new points are gathered. Due to the computation amount, a special contact cache is used for saving the points. The

cache must be updated during the simulation. After each time step a new contact cache is created from scratch. Cache queries are performed as contact points are generated. First, the cache is queried for the interaction pair. If the pair is found, then each contact point entry corresponding to the pair is scanned. If a match is found, the data are retrieved from cache and passed to the solver. The process of the contact cache is described as follow.

Algorithm Contact caching

While *simulate process* ***do***

Generate contact points

For *all contact points* ***do***

query the cache for i

If *cache hit then*

Read points information from cache

Else

Put new information into cache

End If

End For

For *all contact constraints* ***do***

Call for constraints solver to resolve the constraints

End For

End While

IV. OPTIMIZATION FOR REAL-TIME PHYSICAL SIMULATION

In VR environment, the motion and behavior of virtual objects are required to be real time. For real-time physical simulation, the stabilization and performance are especially important. When the forces are balanced, the object will be at rest or moving with constant velocity. Stabilization and performance optimization are equivalent, because the faster the system is stabilized, the fewer iterations the solver needs in processing. Once the system is stabilized the optimization module will turn into sleep state.

The propagation model for contact may require many times of iteration to produce visually appealing results especially in simulations with stacks of objects. If not enough iterations are used, the objects will penetrate into each other. To alleviate this effect, a new shock propagation method is proposed. All the objects in the bottom are assigned infinite mass; the mass matrix is set to zero. If an object of infinite mass is found to be in contact with a higher-level object, its motion is not affected by the impulses. Once assigned infinite mass, objects retain this mass until the shock propagation phase has completed. Consider the stack of objects, starting at the bottom of the stack, each object has its velocity set to zero and its mass subsequently set to be infinite.

Because the resolve process of the physical based simulation is time-consuming task, even the power of nowadays computer is much faster than ever. On the other hand, the interactivity property is an important demand of real time scene building system. The sleeping policy is adopted in the design and implementation of the physical library. A sleepy object is an object that the kinetic energy is zero or the current state implies that it will not move in the immediate future. The benefit of sleepy policy is that there is no need to compute their motion. We can save computational resources by simply doing nothing for sleepy objects.

The simplest method to determine sleepiness is to query the position and orientation of an object [21]. An object is flagged as sleepy whenever its kinetic energy has been zero within a numerical threshold over a specified number of iterations specified by a user. A scale-problem appear because large objects might be handled correctly, but small objects of the same scale as the threshold may be turned sleepy, even though they are moving. Another shortcoming comes from really slow moving objects. These could be turned sleepy if their motion is smaller than the threshold. Another often used heuristic is to track the linear and angular velocities of an object[22]. If the velocities are bellowed the specified threshold then the object turned sleepy. The drawback of the scheme is that linear and angular motions should be handled separately. Another method to determine when objects should be set to sleepy is depending on the total kinetic energy [23-25]. The genius of this is that a natural kinematic weighting occurs of linear and rotational motion, combining them into one measure. This differs from previous approaches which use two threshold tests, one for the norm of the linear velocity and one for the norm of the angular velocity. Using the kinetic energy allows an end-user to set a single threshold value, implying that any object with a kinetic energy less than this threshold is set to sleepy. Thus, the kinetic energy approach overcomes the problem of treating linear and angular motion separately. However, it may suffer from the problem of slow-moving objects, which necessitates tracking over several iterations. We make extension to the traditional approach of sleeping policy. Mass dependent threshold values. Kinetic energy is scaled by the inertia/mass properties of an object, so it makes sense that the absolute threshold test takes this into account. If not, objects with different mass properties will be treated differently.

Combined, all these rules work fairly well in our test application. In the following section we demonstration the extended sleeping policy used in dynamic simulation. Although it seems attractive to apply a sleepy policy in the hope of obtaining real-time simulation, this is not a profitable approach. In real-time applications the user interacts with the world in a dynamic way, and he may thus put everything in motion.

V. IMPLEMENTATION AND RESULTS

The VR Scene Studio is based on the Massive 3D Database. The model includes point cloud model, mesh model, voxel model, analytical model. There are almost 60,000 models in the database. About 1/3 is the model produced from scratch. Other is collected from Internet. The number of Precision models is1023 and high quality models are 6383.With The VR Scene Studio, 3D virtual scene can be easily constructed in short time.

We have developed an application to test our physical library in different scenarios. The 3D model that appears in the scene is acquired from our 3D model database. The dog model is composed by 33854 triangles and 101520

vertices. The terrain model is composed of 115200 triangles. The applications use the OSG to draw the scenes. Figure 6 is our Scene Building System, where the left frame of window is used for choosing 3D models from database, the most right frame of window is especially for setting physical properties, and the middle window is the scene. Figure 7 is the interface to set physical properties. Figure 7 shows the default value of the physical properties. We can add 3D models for rapid scene building in the middle window.

The test machine is a PC with an Intel Core 2 Quad, Q9400 (2.66 GHz), 4 GB of RAM memory and a NVIDIA GeForce 9300 GE with 256 MB of DDR2 dedicated memory. The operating system is Microsoft Windows XP Professional (32 bits). VR Scene Studio is platform of scene building which written by C++ language. The physical library was composed of four distinct modules, which are Inter module, Dynamic module, Constraint module and Optimization module. 1) The Inter module is useful for user to control the properties of objects and interactive with other modules. 2) The Dynamic module implements mathematical functions and responsible for resolving the dynamical equations. 3) The Constraint module is used to resolve constraints effectively. 4) The Optimization module is responsible for ensuring the stability and real-time.

Figure 8 shows several kinds of examples from complex model to stack objects with physical property in our scene building system. Figure 9 is the application of boxes stacked. The restitution coefficient is set to 0.8.The large coefficient of restitution causes simulation errors to be propagated between neighboring boxes. Gray objects indicate sleepy objects. The combination of large restitution and simulation errors cause the box stack to blow-up seen from Figure 9. From equation (3) , impulse in every time step can be computed, while inequality(4),(7) guarantee the non-penetration and the existence of friction. The result showed the physical correctness and the real time property can be achieved at the same time. We test the correctness and real time property by the following application.

Table 1 presents a result for the scenes illustrated by the screenshots in this section. The maximum, minimum and average of rendering time are shown. It showed that the physical library gained the interactive speed.

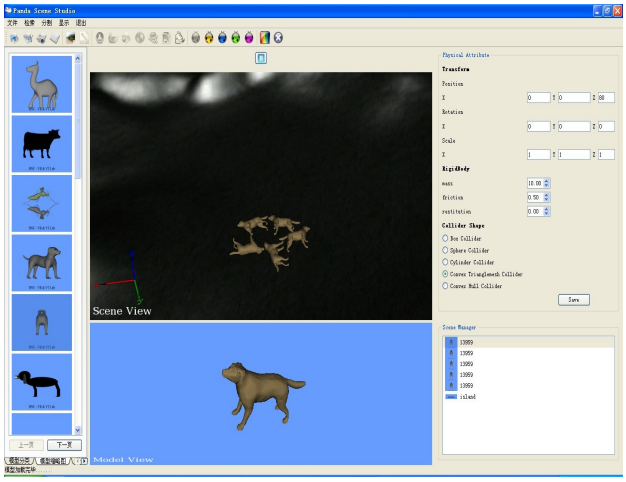

Figure 6. VRStudio -The Scene Building System

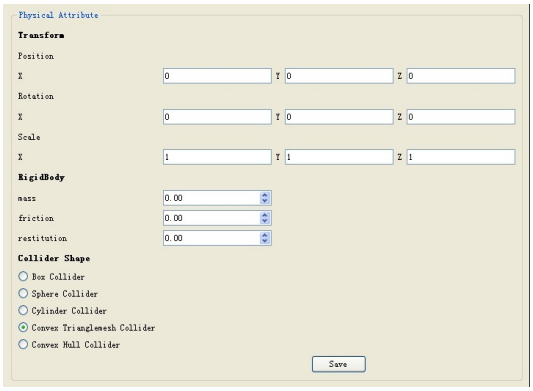

Figure 7. The interface of setting physical properties

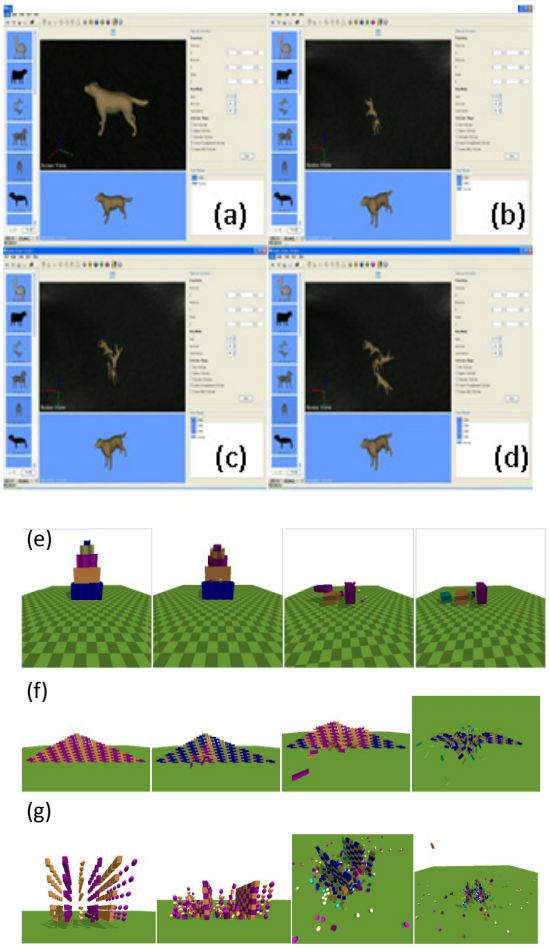

Figure 8 (a) 3D models added to the Scene. (b), (c) and (d) show the dynamic process of the models in gravity field. (e) The stacked cubes are hit by a ball. (f) An arranged cube wall (composed of 210 cubes), some of cubes are moved to destroy the stable state and the correct responses are shown. (g) Plenty of primitive 3D objects (343 primitive objects of sphere, cube and column) released into the air with collision and response.

TABLE I.
TIME USED FOR DIFFERENT SCENES

Scene	Max (ms)	Min (ms)	Average (ms)
Figure 8(e)	0.757	0.013	0.519
Figure 8(f)	28.184	1.041	16.501
Figure 8(g)	29.027	3.354	10.348

To verify the accuracy and efficiency of the algorithms adopted in this paper, we designed another more complicated example to testify whether the interactive response time can be achieved. Figure 10 gives the illustration of the dynamic process of more complicated stacked scene. As our simulation results presented in this section indicate, quite complex and challenging simulations can be done with our optimized impulse-based approach. However the results also show that large scale stacking and many bodies dynamics process are still numerically challenging for the solver.

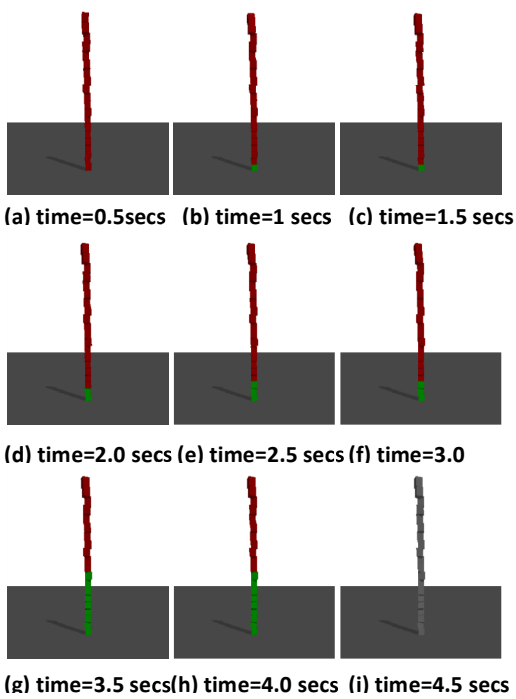

Figure 9 Box-stack simulation with high restitution coefficient of 0.8. Red, green and gray separately indicate active , wants to deactivate and sleepy objects.

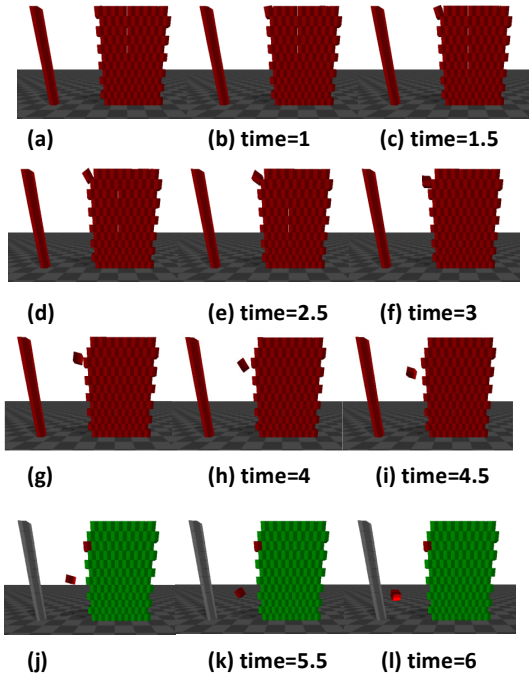

Figure 10 More complicated stacked scene

VI. CONCLUSION

In this paper we presented the design and implementation of a physical module which can enhance the 3D scene building system with more realism. Our contribution can be summarized as follows: the definitions of physical properties are introduced by our physical library, which can be interpreted by the Dynamic module. The core Dynamic module is impulse-based paradigm, and the dynamic equation is established by Lagrange method. An effective contact resolver is proposed in this paper to handle contact constraints effectively and efficiently. Improved shock propagation is also used to improve the stability of the physical simulation.

This work can be readily extended: one promising topic to be explored is GPU parallelization. Internal algorithms or even whole sections of our physical module could be redesigned in order to perform tasks in parallel. With the advent of GPGPU languages and their growing support and popularization, optimizations of this kind have become massively explored by recent works in physics simulation.

REFERENCES

[1] Y Huang, Z Deng, L Yao. Dynamic analysis of a rotating rigid-flexible coupled smart structure with large deformations. Applied Mathematics and Mechanics 28 (10), pp. 1349-1360. 2007. http://dx.doi.org/10.1007/s10483-007-1008-z

[2] COUMANS, E. Bullet physics library. http://bulletphysics.org. 2006.

[3] SMITH, R. 2001.Open dynamics engine. http://www.ode.org.

[4] Liu, X., Du, H., Wang, H., Yang, G. Design and development of a distributed Virtual Reality system. In: International Conference on Machine Learning and Cy-bernetics. vol. 2, pp. 889-894. 2009.

[5] Braun, H., Hocevar, R., Queiroz, R., Cohen, M., Moreira, J., Jacques, J., Braun, A.,Musse, S., Samadani, R. VhCVE: A Collaborative Virtual Environment Including Facial Animation and Computer Vision. In: Games and Digital Entertainment. pp.207-213. 2010.

[6] Physics for Game Developers, 1st ed. O'Reilly Media, Inc., November. 2001.

[7] Morgan Kaufmann.2007.Game Physics Engine Development, book & cd-rom 1st ed.

[8] K Erleben. Stable, Robust, and Versatile Multibody Dynamics Animation. Doctoral Thesis, University of Copenhagen, Denmark. 2004.

[9] Morgan Kaufmann. Game Physics, Second Edition, book &cd-rom 2st ed. 2010

[10] Realistic animation of rigid bodies.1988. In SIGGRAPH'88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques, ACM, New York,NY, USA, pp.299–308.

[11] MIRTICH, B., AND CANNY, J. F..Impulse-based simulation of rigid bodies. In Symposium on Interactive 3D Graphics, 181–188, 217. Pp.1995.

[12] MIRTICH, B. V.Impulse-based Dynamic Simulation of Rigid Body Systems. PhD thesis, University of California at Berkeley. 1996.

[13] Tan Xiao-hui，Fan Ya-chun，Wang Xue-song，Zhou Ming-quan. The Model and Simulation of Non-smooth Chained Multi-body Dynamics.Vol.24(9),pp.1757-1761. 2012.

[14] Y Huang, Z Deng, Y Xiong. High-order model and slide mode control for rotating flexible smart structure. Mechanism and Machine Theory 43 (8), pp.1038-1054. 2008. http://dx.doi.org/10.1016/j.mechmachtheory.2007.07.005

[15] David Baraff. Dynamic Simulation of Non-Penetrating Rigid Bodies. PhD thesis, Department of Computer Science,Cornell University, March. 1992.

[16] Paul U. Lee,Diego C. Ruspini, and Oussama Khatib. Dynamic simulation of interactive robotic environment. In International

Conference on Robotics and Automation.pp.1147-1152.IEEE.May. 1994.

[17] J. Thomas Ngo.Joe Marks. Spacetime constraints revisited. SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques,pp.343-350. 1993.

[18] Vivek Bhatt and Jeff Koechling. Classifying dynamic behavior during three dimensional frictional rigid body impact. In International Conference on Robotics and Automation. IEEE. 1994.

[19] Brian Mirtich and John Canny. Impulsed-based dynamic simulation. In K.Gold-berg,D. Halperin, J.C. Latombe, and R. Wilson, editors, 1994.

[20] W.J.Stronge. Unraveling paradoxical theories for rigid body collisions. Journal of Applied Mechanics.pp.58:1049-1055,December . 1991.

[21] En Bezel. Physically-based Modeling for Computer Graphics, a structured approach. Academic Press. 1992.

[22] Brian Mirtich. Hybrid simulation: Combining constraints and impulses. Proceedings of First Workshop on Simulation and Interaction in Virtual Environments, July. 1995.

[23] Victor J. Milenkovic and Harald Schmidl. Optimization-based animation. SIG-GRAPH Conference. 2001.

[24] Harald Schmidl. Optimization-based animation. PhD thesis, Univeristy of Miami. 2002.

[25] Harald Schmidl and Victor J. Milenkovic. A fast impulsive contact suite for rigid body simulation. IEEE Transactions on Visualization and Computer Graphics,10(2),pp.189–197. 2004. http://dx.doi.org/10.1109/TVCG.2004.1260770

AUTHORS

Tan.Xiaohui is with Institute of Virtual Reality and Visualization Technology, Beijing, Beijing Normal University, China (xiaohuitan@163.com).

Li.Shanshan, is with Institute of Virtual Reality and Visualization Technology, Beijing, Beijing Normal University, China (lss_journey@163.com).

Zhou.Mingquan is with Beijing Normal University, Beijing,100875,China.(mingquanzhou@bnu.edu.cn)

This work was supported in part by National Natural Science Foundation of 61170203 , National Natural Science Foundation of 61104035. Submitted, 16 July, 2013. Published as re-submitted by the authors on 12 October 2013.