PHYSICALLY BASED MODELING IN VIRTUAL ASSEMBLY

Yong Wang, Sankar Jayaram, Uma Jayaram
Washington State University
Pullman, WA 99164

Kevin Lyons, Peter Hart
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

Virtual assembly is a promising application of virtual reality in design and manufacturing that has drawn much attention from industry and research institutes. Physically based modeling has been an important research topic in computer graphics and virtual reality. In this paper, physically based modeling issues in virtual assembly are investigated. The specific requirements and characteristics of physically based modeling in virtual assembly versus those in traditional computer graphics are analyzed and studied. The mass properties of the assembly models are extracted from the Computer Aided Design (CAD) system while the design models are transferred from the CAD system to the virtual assembly environment. The assembly models are categorized using human strength survey data. The interaction between the parts, the environment objects, and the human are analyzed. In the fully immersed virtual environment, it is discovered that the gravity acceleration needs to be scaled down to achieve maximum realistic feeling. Finally, the benefits and limitations of physically based modeling in virtual environments are discussed.

1. Introduction

Virtual Reality (VR) is a synthetic environment that gives a person a sense of reality, even though the visual images of the environment may or may not exist in real life. During the last few years, engineers and researchers have been trying to combine VR technology with traditional CAD/CAM systems. One important use of VR in engineering is the implementation of "Virtual Assembly". VR can play an important role in predicting and representing the physical realization of the product that involves humans. A VR assembly environment called "Virtual Assembly Design Environment" (VADE) has been designed and implemented at Washington State University to represent actual assembly processes [Jayaram 97]. Using VADE, the designer is immersed in a virtual environment where assembly evaluations can be performed. CAD assemblies are obtained from *ProEngineer*™ developer's toolkit along with assembly constraints. These assembly constraints are used to simulate physical constraints in VADE.

In the immersive environment, the behavior of the objects should be "physically correct". This introduces the physically based modeling problems of computer graphics, especially interactive dynamic simulation. This topic has been extensively studied in the computer graphics community for several years. However, most of the research is focused on the simulation of non-penetrate rigid bodies. In the spectrum of physical systems, if one end consists of highly constrained jointed linkages, and the other end free unconstrained moving particles or objects, then the middle consists of interactive systems, which have

been least studied [Mirtich 96]. This is because no existing simulation paradigm is suitable for a system where human beings are involved directly. How to relate the interaction to the motions of the object when the human is directly involved and how to deal with the human's reaction to the interactive system remain difficult issues to be studied.

In this paper, we investigate the methods of dynamic simulation and discuss dynamic behaviors of objects in the virtual environment that can contribute to the assembly design evaluation and analysis operations. We also analyze the interaction in the virtual assembly environment.

2. Literature Review

Much of the research dealing with interactive dynamic simulation in computer graphics has been devoted to contact modeling. Interactive dynamic simulation can be conceptualized using two approaches, constraint-based contact models [McKenna 90, Witkin 90, Baraff 92, Baraff 94, Pong 95, Trinkle 95] and impulse-based models [Mirtich 96]

Physically based modeling cannot be separated from collision modeling. Comprehensive surveys of collision detection algorithms were conducted [Mirtich 96, Gottschalk 96] and several successful collision detection packages were developed [Cohen 95, Gottschalk 96].

3. Extracting Physical Properties from a CAD System

In physically based modeling, the basic equations of motion for rigid bodies used to set up the simulation model are the Newton-Euler equations, which require the mass properties(i.e. mass and inertia matrices) of the part or the system.

Mirtich proposed some fast and accurate methods of calculating mass properties of polyhedral objects [Mirtich 96]. In our system, we get around this trivial problem by getting the information directly from the CAD system when the model is designed. The mass properties are defined (unless the object is broken or deformed) once the model is designed. This information can be extracted using the CAD system (e.g. ProEngineer^{TM} developer's toolkit). When the model geometry and constraint information are written, the mass properties are written to a property file for each part (or assembly) of the model. In the property file, additional information is available including units, surface finish, tolerance values, surface area, density, and volume.

4. Assembly Model Categorization

Assembly models differ tremendously in terms of size and numbers of parts, from tiny motors to large aircraft. Human interaction varies with different assembly models. For some small assemblies, assemblers may use their bare hands with assistance from tools. For large assemblies, they depend on tools, e.g. hoists, to lift some big parts and put them in their final locations.

In the virtual assembly environment, we need to take this fact into consideration. It is easy to use one hand to grab and lift a several hundred pound truck engine in the virtual environment. But this results in the loss of feeling of realism, or even trust of the system. So we need to distinguish and categorize the assembly models according to human behavior and ability.

The criterion we use is the strength survey data of human beings [Diffrient 93]. In our virtual assembly environment, we organize parts into three categories determined by weight: part able to be lifted by one hand, part able to be lifted by two hands, or part able to be lifted by a tool. Although this categorization is crude and simple, it can represent the real world situation. One interesting observation is that novice users tend to reach out their hands to pick up relatively small parts even before any explanation is provided on how to grab the parts in the environment. If a user is put into the environment with a large part in front of them, the user usually stays static and waits for instructions.

5. Dynamic Simulation of Parts in Virtual Assembly

The simple categorization of the parts in the assembly models can help us define the scope of dynamic simulation of the parts in the virtual environment. We explore dynamic simulation only in cases where the models are small and the parts are not heavy, i.e. part able to be lifted by one hand. For larger models and parts, it is not applicable since these kinds of behaviors and motions are not allowed in real industrial world anyway because of safety concerns.

During the assembly process planning of operation, certain behaviors such as object bouncing are not our major concern because we can always assume the user will behave rationally in the assembly operation. What we really care about are the behaviors of the part in the user's hand and on the base part. In the virtual environment we mainly concentrate on dynamic behaviors of the part while the part is held in the user's hand and while the part is constrained on the base part. We study three kinds of motions of that are closely related to virtual assembly modeling: free motion in space, translation on a plane and along an axis, and rotation about an axis.

5.1 FREE MOTION IN SPACE

Free motion in space of an object is the simplest to handle. The object just follows a ballistic trajectory as described in elementary physics texts. The equations of motion are shown in equations (1a) and (1b). Also "G" is not defined in the following explanation. In the equations, **t** is the time of motion, $\mathbf{V_0}$ and $\mathbf{w_0}$ are the initial linear and angular velocity vectors, **S0** and **S** are initial and instantaneous position vectors, and finally, $\mathbf{Ang_0}$ and **Ang** are initial and instantaneous angle values of the object's local coordinate system relative to the global coordinate system.

$$\mathbf{S} = \mathbf{S0} + \mathbf{V_0} * \mathbf{t} + \mathbf{0.5} * \mathbf{G} * \mathbf{t}^2 \quad (1a)$$

$$\mathbf{Ang} = \mathbf{Ang_0} + \mathbf{w_0} * \mathbf{t} \quad (1b)$$

Since we only need to obtain position and orientation values to update our display and the values can be computed directly with equations (1a) and (1b), we do not need to do any integration. The critical issue here is how to obtain the initial linear and angular velocity vectors.

Before the object can move freely in the global space, the part is either held in the user's hand or constrained on the base part. For simplicity, we always keep track of the object's global positions and orientations with respect to time no matter where the object is. The position and orientation information of the object can be represented by a transformation matrix. At the moment when the object is able to move freely in space, we choose two neighboring instances (we call the object in a certain frame an "instance") and calculate the initial velocity vectors based on the differences of positions and orientations of those two instances. This is just an approximation method since we are using finite translation and rotation displacement to calculate velocities. However, since the time difference between two neighboring

frames is usually very small, the approximation can still provide a good initial condition for the free motion of the object.

5.2 SLIDING ON A PLANE OR ALONG AN AXIS

When the part is constrained on the base part, if the part is constrained to move on a plane, the part can only slide on the plane. If the part is constrained by two parallel axes, or one axis and a plane parallel to the axis, the part is only allowed to slide along the axis. If the part is constrained on two planes, the part can only move along the intersection line of the two planes (Figure 1). We need to remember that the base part is manipulated in the left hand of the user and the movement direction of the part may be changing with respect to the global frame.

Figure 1: Object sliding on a plane or sliding along an axis

For these situations, we set up a vector called "**AllowableDir**" to represent the allowable translation direction (see Figure 2). This vector is computed for different situations using equation (2a) or equation (2b). For axis movement, "**AllowableDir**" is along the axis. For a plane movement, "**AllowableDir**" is a vector that is perpendicular to the plane's normal vector.

$$\textbf{AllowableDir} = \text{end2} - \text{end1} \tag{2a}$$

$$\textbf{AllowableDir} = \mathbf{n} \times (\mathbf{G} \times \mathbf{n}) \tag{2b}$$

The symbol "×" represents the cross product of two vectors. If the angle between **G** and **AllowableDir** is greater than 90^0, we negate **AllowableDir**.

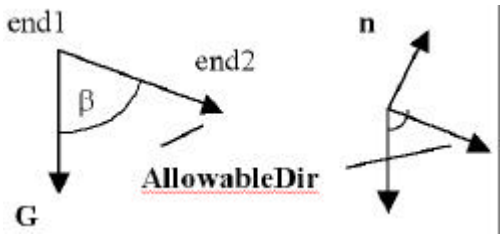


Figure 2: **AllowableDir** *computation. Here,* end1 *and* end2 *are the two end points of an axis,* **n** *is the normal vector of a plane and* **G** *is the gravity acceleration vector. All are represented in the global coordinate system.*

Even if it has a direction to move, the part may not be able to move if we take static friction into account. Suppose the static friction coefficient between the part and the base part is fs; the condition for the part to be able to start moving is checked with equation (3).

$$\beta < 90 - \tan^{-1}(fs) \tag{3}$$

After the motion begins, the dynamic friction coefficient, fd (which is smaller than fs), is used to get the acceleration **a** by equation (4a). In this equation, β is the angle between **G** and **AllowableDir,** m is the mass of the object, and |**G**| is the magnitude of **G**.

The equations of motion (4) are applied to this situation. In these equations, **a**, **V** and **P** represent the acceleration, velocity and position of the object. Notice that **AllowableDir** is changing with the movement of the base part, the position of the part is actually obtained by simple numerical integration using Euler's method:

$$\mathbf{a} = F/m = (m|\mathbf{G}|\cos(\beta) - fd*m|\mathbf{G}|\sin(\beta)) / m * \mathbf{AllowableDir}$$
$$= (|\mathbf{G}|(\cos(\beta)-fd*\sin(\beta))*\mathbf{AllowableDir} \quad (4a)$$
$$\mathbf{V_{n+1}} = \mathbf{V_n} + \mathbf{a} * \mathbf{t} \quad (4b)$$
$$\mathbf{P_{n+1}} = \mathbf{P_n} + \mathbf{Vn} * \mathbf{t} + \mathbf{0.5*a*t^2} \quad (4c)$$
$$\mathbf{dP} = \mathbf{P_{n+1}} - \mathbf{P_n} \quad (4d)$$

The vector **dP** will be used to form a transformation matrix to update the position of the part.

5.3 ROTATING ABOUT AN AXIS

If the part is constrained on the base part by an axis, the part will tend to rotate about the axis if the center of mass of the part is not on the axis. In this case, first, we calculate vector **RotVec** (the vector and object is rotating about) and **CMVec** (the vector that passes the center of mass and perpendicular to **RotVec** from equation (5) and equation (6).

$$\mathbf{CMVec} = (end2-end1) \times ((CM - end2) \times (end2 - end1)) \quad (5)$$
$$\mathbf{RotVec=G \times CMVec} \quad (6)$$

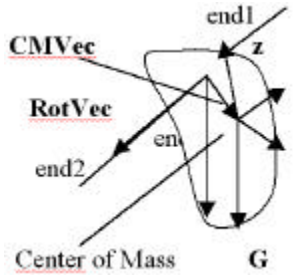


Figure 3: Rotation about an axis. Note that there is a local coordinate system with the origin at the center of mass (its orientation is defined when the part is designed in the CAD system)

In this particular situation, Euler's equation can be simplified to equation (7), where Jaxis is the moment of inertia of the object with respect to the axis of rotation, **w** and **a** are the angular velocity and the acceleration, m is the mass, |**G**| and |**CMVec**| are the magnitudes of vectors **G** and **CMVec**, β is the angle between vector **CMVec** and **RotVec** shown in Figure 3, and fr is the rotational friction coefficient. For simplification, the frictional torque is represented as fr***w.**

$$\mathbf{a} = \text{Torque} / \text{Jaxis} = (m * |\mathbf{G}| * |\mathbf{CMVec}| * \sin(\beta) - fr*w) / \text{Jaxis} \quad (7)$$

In equation (7) **G** is a constant vector, **CMVec** and β can be easily calculated, m can be queried from the part directly, so the only thing left is to compute Jaxis. In Section 3 we described the extraction of mass properties of the part, including the inertia matrix with respect to center of mass, **Icm**. **Icm** is a second order tensor. To calculate Jaxis from **Icm, t**he first create another coordinate system, with the origin still at the center of mass, and one axis (the **z** axis**)** parallel to **RotVec** (Figure 4). The next step is to find a transformation matrix that relates the two coordinate systems. This matrix, **T**, can be formed by rotating **z** to **z'**. The new inertia matrix of the object, **Icm'**, with respect to the new coordinate system x'-y'-z' can be obtained by Equation (8).

$$\mathbf{Icm'} = \mathbf{T} * \mathbf{Icm} * \mathbf{T}^t \quad (8)$$

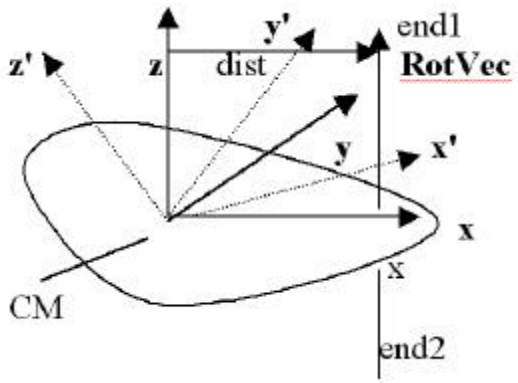


Figure 4: Compute Jaxis. **x-y-z** *is the original coordinate system with* **Icm**, **x'-y'-z'** *is the new coordinate system with z' parallel to* **RotVec.**

Let **z'** be the axis of rotation; the moment of inertia about **z'** is just Iz'z', or **Icm**[2]. Using the general parallel axis theorem, with dist calculated from equation (9), Jaxis can be computed from equation (10).

$$\text{dist} = ((\text{end1}-\text{end2}) \times ((\text{end1}-\text{end2}) \times (\text{CM}-\text{end1}))) \bullet (\text{CM}-\text{end1}) \quad (9)$$

$$\text{Jaxis} = Iz'z' + m*\text{dist}^2 \quad (10)$$

With Jaxis computed, we can form equations (11) to integrate the rotation angles of the object about **RotVec**. In equation (11a), α is the angular acceleration computed in equation (7) while $\mathbf{w_n}$ and $\mathbf{A_n}$ are initial angular velocity and angles for each integration step.

$$\mathbf{wn+1} = \mathbf{w_n} + \alpha * \mathbf{t} \quad (11a)$$

$$\mathbf{An+1} = \mathbf{A_n} + \mathbf{wn} * \mathbf{t} + \mathbf{0.5}*\alpha * \mathbf{t}^2 \quad (11b)$$

$$\mathbf{dA} = \mathbf{A} - \mathbf{A_0} \quad (11d)$$

Finally, **dA** is used to form a rotation matrix to adjust the old part transformation matrix in of the part in the global space.

In equation (4) and equation (11) we adopted the most approximate integration method (according to [Kreyszig 93]). Although its accuracy is of **O**(h2), practically it is good enough. The reason is that the absolute positions and angles are not critical and the approximation is enough as long as the motion looks correct. For example, we will not be able to notice a difference of several degrees in the virtual space when a part is rotating. If the part follows a pendulum like motion, we will tend to believe the motion is correct.

6. Collision Detection and Interaction Analysis

When the part moves in space, or moves on the base part, we want the part to stop moving if its motion is blocked, e.g. stopped by the table or stopped by the base part geometry. As we mentioned before, our main purpose of simulating dynamic behaviors is to assist assembly operation and we do not pay much attention once the part is out of the user's hand or is away from the base part. For the former case, we let the part stop moving if the part hits the table or other environment objects and we do not go further to figure out the balanced resting position or orientation for the part on the table. This saves computation time and lets us concentrate on interaction issues.

The situation is complicated if the part moves on the base part. This situation is illustrated in Figure 5. The part is sliding on a plane P1 on the base part. If the part moves in the direction of **t1**, we need to use collision detection to check whether the part is still touching the base part. If the part slides away from the base part, it goes to free space. If the part moves along **t3**, we need to use collision detection to check if the part will be blocked by P2. If the part moves along **t2**, we do not know which situation will occur first so we need to check for both situations. This brings up a conflicting collision problem- a simple report of whether the part is colliding with the base part is not enough. If the part is moving along **t1** we need always to detect collisions of the two touching surfaces; if the part is moving along **t3**, we must find the collisions of the part other than with P1. The same situation will occur when a shaft is inserted into a hole, the part may be sliding out of the base part or may be blocked by other geometry other than the hole. Usually collision algorithms report the collision status (colliding or not) and the intersecting triangle pairs between two models. Some algorithms even support the option of reporting either the first collision or all the collision pairs [Gottschalk 96].

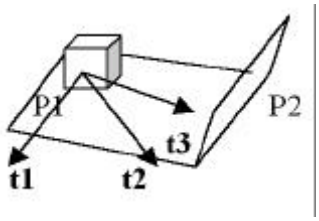


Figure 5: The part can move in any direction on the base part.
P1, P2 *are planes (or geometry) on the base part.*

Since we mainly focus on planar and axial constraints, we add a special set of options for coplanar and coaxial situations in the general collision detection algorithms: "KEEP_COPLANAR" (report if the triangles are parallel and the distance between them is less than a tolerance value), "SKIP_COPLANAR" (do not report if the triangles are coplanar and very close), "SKIP_CYLINDER" (do not report if it is a equal-diameter coaxial insertion), "NOCARE_COPLANAR" (normal mode, do not consider coplanar or coaxial problems). When the part moves on the base part, we perform two collision detection checks: one

for checking if the part is still touching the base part and another one for checking if the part is blocked by geometry of the base part other than the plane or the cylinder the part is sliding on. The first check always detects the collision if the part is touching the base part, so we use the option "KEEP_COPLANAR". The second check always ignores the collision between the touching triangles; we use "SKIP_COPLANAR" or "SKIP_COAXIAL". In Figure 5, if the part moves along **t1**, the first check tells us whether the part still touches the base part. If the part moves along **t3**, the second check notifies us whether the part is blocked by P2. If the part moves along **t2**, whichever of the two checks is first puts the part either in space or on the base part.

With the collision detection problem solved, we can analyze the interaction issues in the virtual assembly environment where we have the user, the part, the base part, the tools, and the environment objects. Since we are investigating how the part is being handled and how the part is being assembled, the part is the center of the whole system. The user can grab or lift the part with their bare hands or with tools, making the user the decisive factor for the part and the tools. If the user uses a tool to manipulate the part, the part should experience feasible motion defined by the tool.

We use different state variables to define the status of a part in the virtual environment. The states are: INHAND (grabbed or lifted by the user), ONTOOL (manipulated by a tool), ONBASESLIDE (constrained on base and can move relative to the base part), ONBASESTATIC (constrained on base and cannot move relative to the base part), INSPACE (free moving in space), STATIC (remaining in the same position and orientation), and PLACED (placed on the base part in the final assembled location). The part will be handled according to these different states. If the part is INHAND, the motion of the part is totally decided by the user's hand. If the part is ONTOOL, its motion is decided by the tool, whose motion is decided by the user. If the part is ONBASE or INSPACE, dynamic simulation methods discussed in Section-5 will be used to figure out the motion of the part. If the part is STATIC, we do nothing. The states of the part can change from one to another. A transition diagram shown in Figure 6 can be used to demonstrate the changes of the state of a part and the cause of these changes. The state diagram also shows the interactions between the user, the part, the base part, the tools, and the environment objects. The diagram provides a convenient way to handle the part in different situations. Whenever an interaction occurs, the state of the part is updated, the part is then handled according to its previous state and the current state.

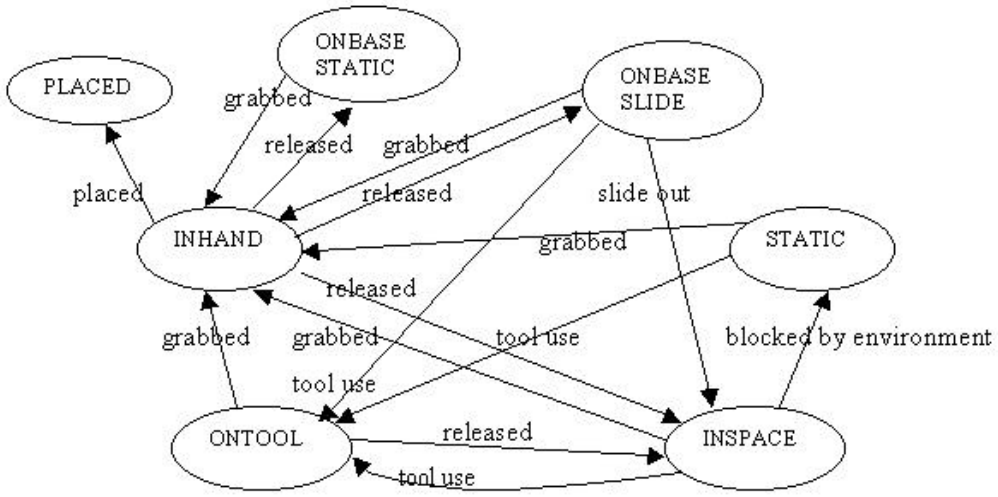


Figure 6: State transition diagram of a part

7. Determining appropriate gravity acceleration

If a part moves freely in space, it will follow a simple ballistic trajectory as in equation (3). Once the initial conditions are known, the only thing that changes the motion of the part is its gravity with acceleration **G**, which is 9.8 meter/second^2 or 385.8 inch/second^2. Surprisingly, to let the user feel that the motion of the part is realistic, we need to scale **G** down to a smaller value using a scale factor 0.2-0.3. The scale factor is attained by trial and error.

Another interesting and conflicting observation occurs in the following circumstance: when we perform the integration for a part that rotates on the base part, we do not need to scale **G** down when we calculate angular acceleration using equation (17) in order to feel the rotation is realistic. The explanation of this phenomenon can be found in the nature of the immersive virtual environment. In real world space, humans usually use their eyes to follow the moving objects and their eyes can follow a moving object even when the object moves with an acceleration greater than that caused by gravity. According to survey by Durlach, et al[Durl 95], human eye movement can be "as fast as 600 degrees/second". But in the virtual space, the human's viewing update is usually determined by the movement of a tracking device attached to their head, instead of their eyes. Even if the graphics of the system can be updated 30 frames per second, the movement of the human's head, especially rotation, is limited ("peak velocities can be about 600 degrees/second for yaw, and 300 degrees/second for roll and pitch"[Durl 95]). In a fully immersed environment, the user usually wears a head-mounted-display device, i.e. a helmet. The weight and inertia of the helmet further limit the motion of the user's head.

Suppose the object is 1 meter away from the user's eyes. If the object is dropping with normal gravitational acceleration as in Figure 7, the head of the user needs to rotate with an angular acceleration computed in equation (12). This requires that the user's head rotate 280 degrees in one second ($0.5 * \alpha_{eye} * t^2$). This is close to the peak ability of head movement (300 degree/second) without a helmet. It is impossible for the user to rotate their head in the virtual space with a helmet on their head to fulfill this requirement.

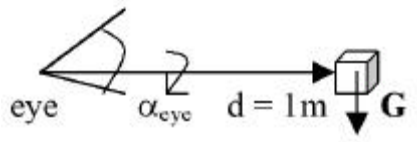


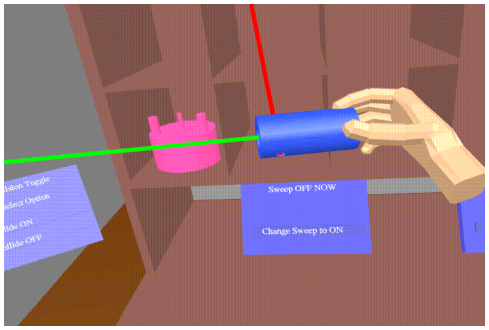
Figure 7: Humans eyes follow the motion of a dropping object

$$\alpha_{eye} = |\mathbf{G}| / d = 9.8 / 1 = 9.8 \text{ (rad/second}^2\text{)} \approx 561 \text{ (degrees/second}^2\text{)} \quad (12)$$

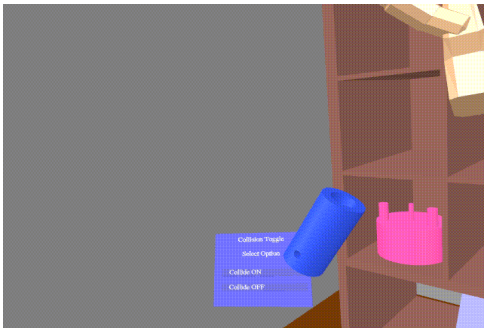
If we recall the scale factor we tried out, which is about 0.25, the motion requirement becomes 280*0.25 = 70 (degrees) in one second, which is a quite reasonable number. Therefore, we can find that the appropriate gravity acceleration in virtual space is determined by the human factor, i.e. the ability of a human's movement in the virtual environment. This also can explain why the gravity acceleration need not to be scaled down for rotating objects: the rotation of the object is usually a local motion, the position of the object in space does not change much and the user does not need to move their head to follow the motion of the object.

8. Results

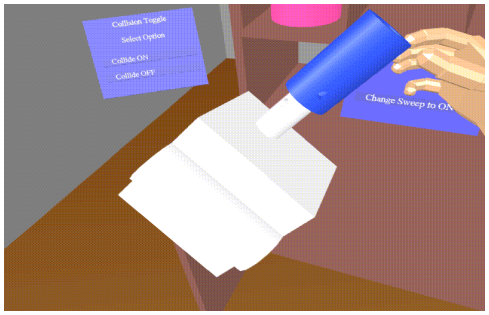
We use some screenshots to illustrate the results. In Picture 1, the cylindrical part is just about to drop down from the user's hand. The object then follows a ballistic trajectory shown in Picture 2. The part is released and constrained on the base part in Picture 3. The part slides down along the extruded cylinder on the base part first and then is blocked by the base part in Picture 4. We then rotate the base part where the blue part is free to move. It slides away from the base part and begins to move in space in Picture 5. We use another assembly model to illustrate the motion of the part while it is partially constrained by a plane. In Picture 6, the part is released on the base part and begins to move on the plane. We tilt the base part and the part begins to slide down, as in Picture 7. Finally, another model is used to illustrate rotational motion. In Picture 8, a part (a gear) is about to be released that is constrained to an axis. The part then follows a pendulum-like rotational motion as shown in Picture 9. In this situation, if the long shaft is tilted, the part will slide along the shaft and still rotate about the shaft, as shown in Picture 10.



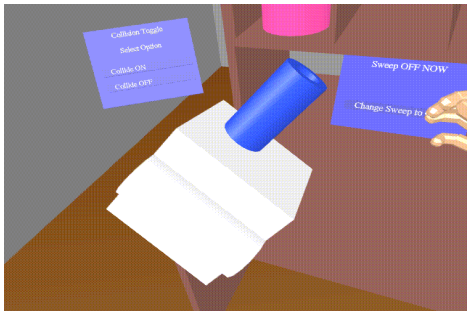
Picture 1: The part is about to drop down from the user's hand



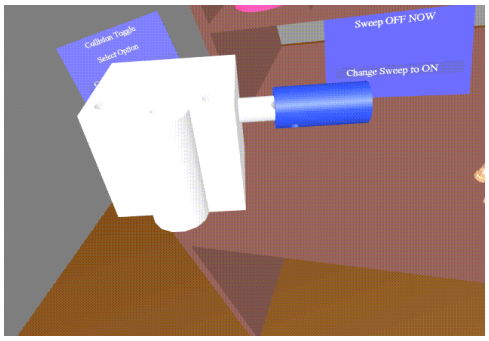
Picture 2: The part drops down because of gravity



Picture 3: A part is about to be released while the part is being constrained. The part will then slide down along the extruded shaft.



Picture 4: The cylindrical part slides down but its motion is blocked by the base part.



Picture 5: The base part (block like) is tilted and the cylindrical part slides down along the cylinder on the base part because of gravity.

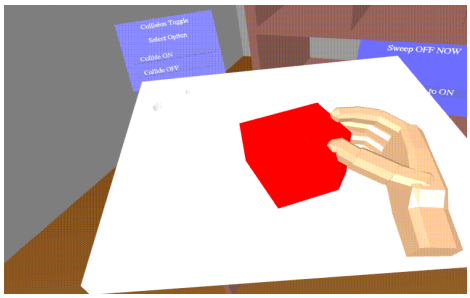
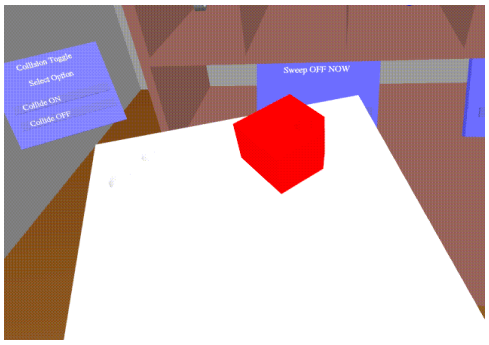
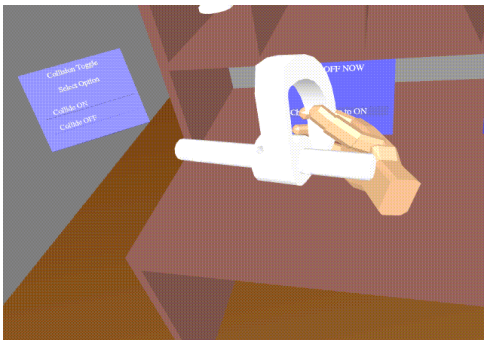


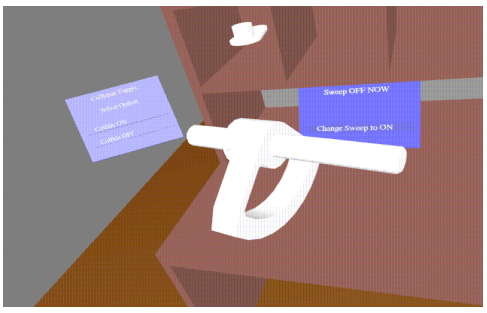
Figure-6: The block part is put on the top planar surface of the base part.



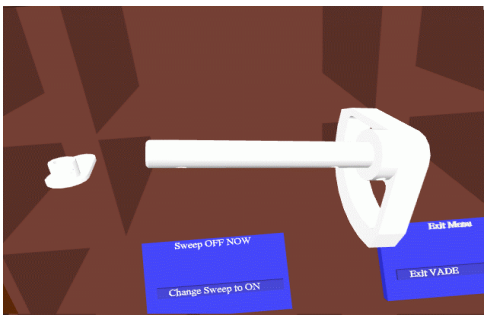
Picture 7: The block part slides down on the plane.



Picture 8: A rotational part is put on a shaft and is ready to be released.



Picture 9: The part rotates about the shaft.



Picture 10: The part is rotating about the shaft and also sliding along the shaft. The shaft is manipulated by the user's left hand.

9. Conclusions

Dynamic simulation of the parts in the virtual environment can greatly enhance the realistic feelings of virtual spaces. Experiments show that after the user finds out that the gravity and inertia are taken care of realistically, they will begin to try various things like throwing the part up and attempting to grab it in space quickly. The trust in the environment continues even when the user exits the virtual environment.

An interesting phenomenon occurs here. After the user begins to perform the assembly operations, the freshness of the dynamic simulation wears off gradually because the user begins to concentrate on the assembly and not the behaviors of the individual parts. The constrained simulation mechanism will guide the user through the assembly process. This leads us to draw a conclusion that dynamic simulation increases the realistic feeling of the virtual environment but does not contribute greatly to the assembly evaluation and analysis in most cases.

One important issue is that this kind of interaction is only feasible, applicable and useful in small assembly models. If the part is heavy or big, it will not be allowed to move freely in real assembly operation. Also, big parts and large assembly models contain a high degree of geometry. The key concern will be how to achieve a higher frame rate that is as fast as possible. Therefore, dynamic simulation is not viable and appropriate for these situations.

The main purpose of physically based modeling in virtual assembly is to assist in simulating the design intent of the designer. Overall, constrained motion simulation is the convenient way to achieve this goal. The constrained motion methodology aligns the concepts of constraints in the assembly design with the actual assembly operation. When we simulate the physical constraints, we not only represent the designer's intent, but also show the processes in which the assembly is physically carried out. This also provides a direct and intuitive method of assembly evaluation since we are simulating the physical assembly process. The simulation can be used for all sizes of assembly models and is computationally effective since it avoids extensive collision checking and motion calculations in every frame.

10 Acknowledgement:

This research was supported by the National Institute of Standards and Technology (NIST), Manufacturing Systems Integration Division.

REFERENCES

[Baraff 92] D. Baraff. "Dynamic Simulation of Nonpenetrating Rigid Bodies", Ph.D. Dissertation, Department of Computer Science, Cornell University, 1992.

[Baraff 94] D. Baraff. "Fast Contact Force Computation form Nonpenetrating Rigid Bodies", In Proceedings of ACM SIGRAPH, 1994.

[Chandrana 97] H. Chandrana. “Assembly Path Planning Using Virtual Reality Techniques”, MS thesis, School of Mechanical and Materials Engineering, Washington State University, May, 1997

[Cohen 95] J. Cohen, M. Lin, D. Manocha, and M. Ponamgi. "I-collide: An Interactive and Exact Collision Detection System for Large-scale Environments", In Proc. of ACM Interactive 3D Graphics Conference, pages 189-196, 1995

[Connacher 95] H. Connacher, S. Jayaram, and K. Lyons. “Virtual Assembly Design Environment”, Proceedings of 1995 Computers in Engineering Conference, Boston, MA, September 1995.

[Diffrent 93] N. Diffrient, A. R. Tilley, and D. Harman. <<Humanscale 4/5/6>>, Henry Dreyfuss Associates, 1993

[Durlach 95] N. Durlach, and A. Navor. “Virtual Reality, Scientific and Technological Challenges”, National Academy Press, 1995

[Gottschalk 96] S. Gottschalk, M.C. Lin, and D. Manocha. "OBBTree: A Hierarchical Stucture for Rapid Interference Detection", In Proc. of ACM SIGRAPH, New Orleans, 1996

[Jayaram 97] S. Jayaram, H. Connacher, S. Jayaram, and K. Lyons, “Virtual Assembly using Virtual Reality Techniques”, Computer-Aided Design, Vol. 29. No. 8 August 1997.

[Kreyszig 93] E. Kreyszig, "Advanced Engineering Mathamatics", John Wiley & Sons, Inc. 1993

[Lin 93] M.C. Lin. "Efficient Collision Detection form Animation and Robotics", Ph.D. Thesis, Department of Computer Science, University of California, Berkeley, 1993

[McKenna 90] M. McKenna, and D. Zeltzer. "Dynamic Simulation of Autonomous Legged Locomotion", Computer Graphics, Vol. 24, No. 4, 1990

[Mirtich 95] B. Mirtich. "Hybrid Simulation: Combining Constraints and Impulses", in Proceedings of First Workshop on Simulation and Interaction in Virtual Environments, July 1995

[Mirtich 96] B. Mirtich. “Fast and Accurate Computation of Polyhedral Mass Properties”, Journal of Graphics Tools, Volume 1, Number 2, 1996

[Mirtich 96] B. Mirtich. "Impulse Based Dynamic Simulation of Rigid Body Systems", Ph.D. Dissertation, Department of Computer Science, University of California, Berkeley, 1996

AUTHOR CONTACT INFORMATION

Sankar Jayaram

Contact information:

School of Mechanical and Materials Engineering
P.O. Box 642920
Pullman, Washington 99164-2920
Email link: mailto:jayaram@mme.wsu.edu

Uma Jayaram

Contact information:

School of Mechanical and Materials Engineering
P.O. Box 642920
Pullman, Washington 99164-2920
Email link: mailto:uma@mme.wsu.edu

Kevin Lyons

Contact information:

Manufacturing Systems Integration Division
National Institute of Standards and Technology
100 Bureau Drive, MS8260
Gaithersburg, MD 20899-8260
Email link: mailto:klyons@nist.gov

Yong Wang

Contact information:

Manufacturing Systems Integration Division
National Institute of Standards & Technology
100 Bureau Dr. Stop 8260
Gaithersburg, MD 20899-8260
Email link: mailto:yong.wang@nist.gov

Peter Hart

No contact information