Special Issue: CAD in the Arts

Kinetic Art Design System Comprising Rigid Body Simulation

Yohsuke Furuta[1,2], Jun Mitani[1,2], Takeo Igarashi[1,3] and Yukio Fukui[2]

[1]Japan Science Technology Agency
[2]University of Tsukuba
[3]The University of Tokyo

ABSTRACT

Kinetic art, such as mobile sculptures and balance toys, is art that involves movement. The design of original kinetic art is difficult because one must consider physics in order to predict the result. Current computer aided design (CAD) and computer aided engineering (CAE) tools are of limited use when designing such dynamic objects because typically these systems are entirely separated. In this paper, we propose a new system for designing original kinetic art objects in which a three-dimensional (3D) geometrical modeling interface and a rigid body simulation are combined. In this study, we introduce an implementation technique to achieve real-time responses and present a prototype system. To demonstrate the technique's effectiveness, we report the results of user studies and show several objects designed on the system.

Keywords: CAD, CAE, physical simulation, rigid body, kinetic art.
DOI: 10.3722/cadaps.2010.533-546

1 INTRODUCTION

Kinetic art, such as mobile sculptures (Fig. 1) and balance toys, (Fig. 2) is art that involves movement. The designed motion can be produced by wind, flowing water or by observers. Numerous works of this type have been created by artists [1][14] since the early 20th century. Movable objects attract people, but designing shapes that move as the designer intends is not always easy. In this paper, we propose a computer system for use when designing “kinetic” art.

The use of computers for designing objects in the real world has become commonplace. A computer aided design (CAD) system is considered imperative for designing industrial products, and designing CAD user interfaces that are suitable for novice users has been a common topic of recent studies [9][18][19]. By using a CAD system and a rapid prototyping system such as a three-dimensional (3D) printer that have dramatically decreased in cost recently, we can easily design actual objects that are the intended shape.

However, current CAD systems can only be used for geometrical modeling and they do not evaluate the movement of designed objects. Even though it is possible to simulate and analyze objects with a computer aided engineering (CAE) system, we have to return to the CAD system if the CAE

results are not satisfactory. Because current CAE systems are usually expensive and require knowledge of numerical analysis in order to master them, they are not easy to use for non-specialists. Furthermore, it is inconvenient to go back and forth between separated CAD and CAE systems, especially since almost all kinetic art is created through trial and error.

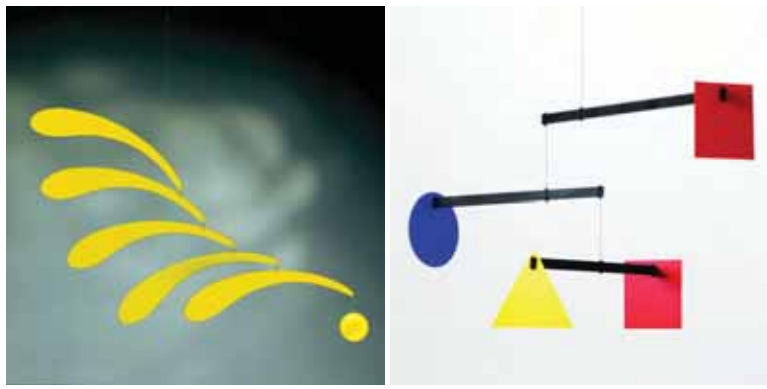


Fig. 1: Examples of mobiles [14].

Fig. 2: Examples of balance toys (Left: Ishiguro Co., Ltd.).

In this paper, a 3D CAD system with rigid body simulation is proposed. To create this system, we combined a 3D modeling application with a rigid body simulation engine in order to aid the design process of kinetic objects. The system continuously computes the motion of objects and presents the results during a modeling operation. This allows the user to quickly execute many trial-and-error experiments. To provide quick response to operations, we present a new scheme in which the system adjusts the tradeoff between accuracy and response time. The motions of the designed objects are visualized as a set of wireframe models; and annotations, such as the colored arrows in the modeling window (Fig. 3(c)-(e)), convey the motion.

Our contributions are summarized as follows:

- We propose a new 3D CAD system in which the rigid body simulation runs continuously during geometric modeling.
- We introduce an implementation technique to provide real-time responses to user operations. The responses are achieved by adjusting the size of the simulation time steps depending on the user operations.

- We introduce a method for visualizing the motions of objects. This method does not use animation; instead, in the modeling window, we show a set of wireframe models along with colored arrows that indicate the predicted motions.
- We developed a proposed application for designing kinetic objects that integrates the methods discussed above. To demonstrate the effectiveness of our approach, we then conducted a user study and created several mobile objects.

In section 2, we introduce previous work related to our study. In section 3 we describe details of our implementation. Then we show the results of our user study in section 4. Several objects designed with our system are shown in section 5. Lastly we summarize results of our study in section 6.

2 RELATED WORK

The graphics community in general has had a long-standing interest in the design and simulation of objects and systems that behave in a physically realistic manner. Mitani [9] developed a system that generates a paper craft toy design from a given 3D model, and Weyrich presented a system for creating bas-relief sculpture [18]. Other studies have developed specialized modeling systems that are capable of designing physical objects from scratch. For example, Mori presented a system for designing an original plush toy by sketching [19]. In our system, we assist the user in designing physical objects by using rigid body simulation and applying the result to the design of kinetic art.

Rigid body simulation is now commonly used in video games, and some recent systems have integrated such simulation into geometric modeling. ASSIST (A Shrewd Sketch Interpretation and Simulation Tool) [2] and Phun [5] integrate rigid body simulation with geometric modeling based on sketches. The user draws shapes of rigid body primitives on the screen, after which the system immediately runs a simulation based on them. SketchyPhysics [6] is a 3D rigid body simulation plugin for Google SketchUp [7]. The user can make interesting and realistic 3D animation easily with the system. Prometech Software developed a similar system named PhysiCafe [16] that combines a geometric modeling interface with a particle-based fluid simulation. However, these systems still require the user to complete the geometric modeling and then view an animation sequence in order to observe the consequences. Our approach pursues tighter integration of simulation and modeling. The system runs a simulation in the background during interactive modeling performed by dragging selected features with the mouse, and continuously shows the resulting motions as visual annotation as the scene is edited. Umetani developed a system named DelFEM [12] that combines an CAD interface with a FEM simulation. We can design the shape of object while viewing the FEM analysis. But the system doesn't support shape dynamically transformed.

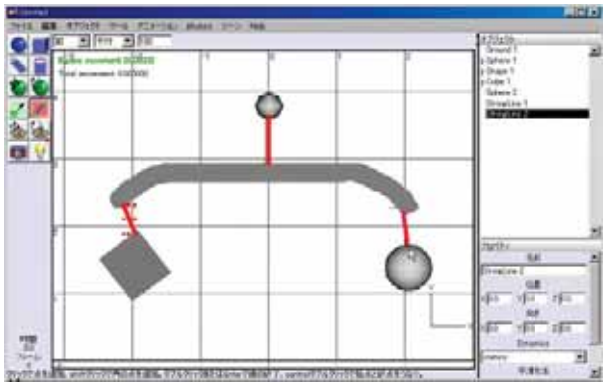
The technique for sacrificing visual quality during user interaction is often used for quick response. For example, Tang et al. proposed a method to get a quick visual feedback by omitting complex objects from rendering [17]. On the other hand, our approach adjusts simulation quality by continuously changing the size of the time step.

Continuous presentation of the results of a rigid-body simulation during direct manipulation was examined by Popovic et al. [8] in their animation authoring system. The user interactively specifies the desired target configuration during which the system runs an optimization to obtain appropriate parameters such as the initial position and velocity. In our proposal, we borrowed the idea of presenting the simulation result as visual annotations and applied this idea to the geometric design of actual physical systems comprising multiple objects.

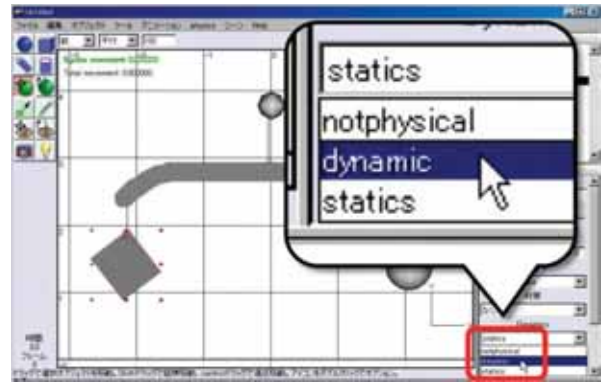
Our work also evolves from various visualization methods for moving images and objects. For example, Goldman et al. generated a storyboard from a video sequence in order to visualize motion [3]. Nienhaus et al. proposed dynamic glyphs that visualize motions of objects in static media [10]. Paths, arrows, and texts are used for representing motions and collisions in the system. We referred to this idea for our 3D dynamic system. Needham et al. visualized the predicted movements of billiard balls on real pool table using a projector [11]. The system analyzes positions of balls and direction of the queue from images and estimates their motions. A common point with our system is that user can adjust the initial position by seeing the simulated results.

3 SYSTEM OVERVIEW

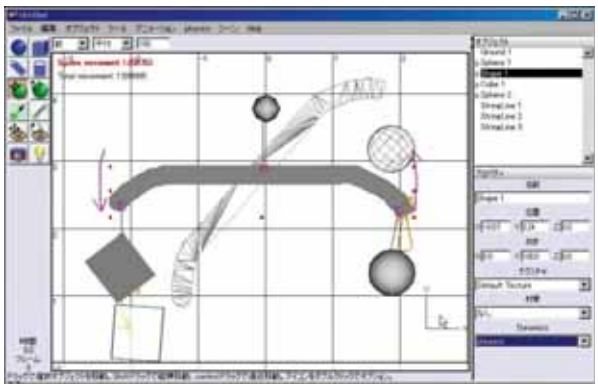
Figure 3 shows an example of a modeling sequence and GUI of our system. The system is an extension of “Art of Illusion 2.6.1” (AOI) [15], a 3D modeling application written in Java whose source code is distributed under the General Public License (GPL). Although this application does not have the advanced functions seen in recent CAD systems, users can design simple objects using its intuitive interface. We first added a new function that connects two objects by a string so that the users can design mobiles or balance toys. We used “NVIDIA PhysX 2.8.1” [13] to simulate the object behaviors in 3D virtual space. This library is a widely used 3D rigid body simulator, and some NVDIA’s GPU can take advantage of it. This library supports many functions like rigid body simulation and collision detection, advanced character control, kinetics for vehicle, fluid simulation, cloth simulation, soft body simulation, force field, and so on. Because AOI is written in Java and PhysX is provided in native code, we modified AOI to call PhysX’s API through Java Native Interface (JNI). And in our system we use the function of rigid body simulation and collision detection of PhysX.



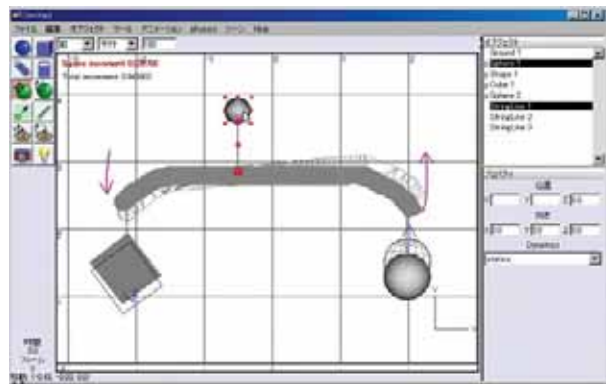
(a) Create shapes and connect them with strings.



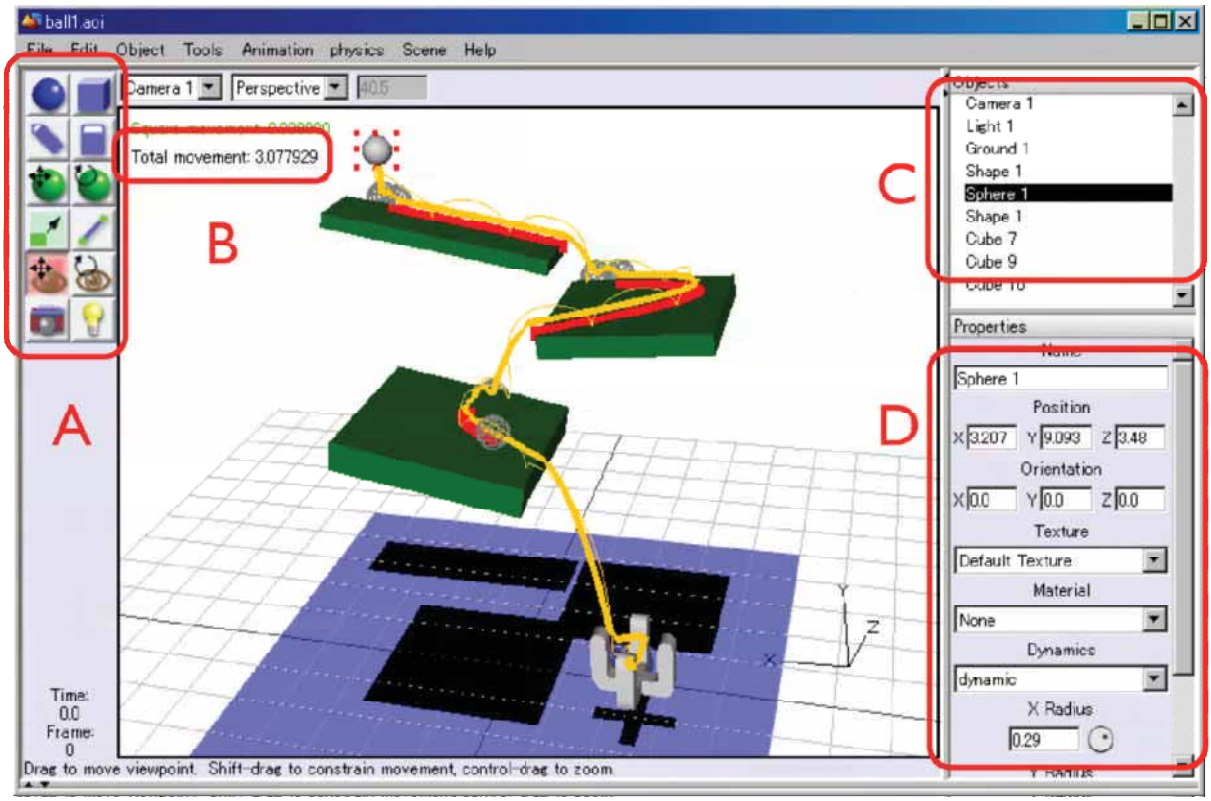
(b) Set shape properties to “dynamic”.



(c) Start a simulation.



(d) Adjust the shapes by observing the simulation results.



(e) Main screen. User edits shapes and positions of objects. The result of physical simulation is superposed. (A) Tool buttons. (B) Amount of total movement of objects. This value is used for evaluation in Section 4. (C) Object list. (D) Properties of the selected object.

Fig. 3: An example of a modeling session and of the user interface of our system.

The user can create spheres, cubes, and flat boards with simple mouse operations in 3D space. The objects are represented as a set of triangles. The user can edit the details of the shape by dragging vertices. Further, the user can connect two objects with a string by a dragging operation. The “total movement,” i.e. the summed up distances of all the vertices moved, is displayed on the screen (this is used for evaluation in Section 4). To realize a modeling system that was tightly combined with a physical simulator, we added properties such as direction, density, and attribution to each object. The attribution has three possible states: “dynamic”, “static” and “nonphysical”. The behaviors of the dynamic objects are calculated by the simulator. The static objects do not move, although they do affect other dynamic models when they collide. The nonphysical objects are temporarily excluded from the simulation. Other properties are shown in Table 1. One of the most important parameters is the size of the time step because this affects both the accuracy of the simulation and the response delay. In the next subsection, we discuss two schemes used to decide the size of the time step.

Object type	*Properties*
Sphere	Radius
Cube	Width, height, and depth
Flat board	Thickness, 2D vertices of stroke
Triangle mesh	3D vertices, triangle indices

Tab. 1: Objet types and their properties and parameters.

The positions and properties of the objects are automatically passed to the physical simulator from the modeling system via JNI. In the simulation, we used the following material properties for all objects: density 1, elasticity 0.4, static friction 0.5, and dynamic friction 0.5.

The results returned from the simulator contain the positions of the objects after a specified time step has past. Collisions between objects can also be detected from the results. For each collision, the simulator returns the collision position and the IDs of the two objects involved in the collision.

3.1 Responsive Physical Simulation

Physical simulation requires significant computational resources. These include not only the cost of the simulation itself (which causes response delays) but also the cost of initialization. Initialization includes the division of objects into convex polyhedra, computing a convex hull, re-indexing the vertices, and so on for improving the speed of the collision tests. To reduce the computational time, we introduced the following schemes:

- We perform initialization only for objects whose shape has been changed.
- We change the size of the time step of the simulation (Δt) dynamically based on the user operation.

The second scheme is based on the concept that when a user is changing the position of an object, a rapid response is more important than the accuracy of the simulation. Conversely, when a user stops an operation, the simulation should be sufficiently accurate because the user is inspecting the result on the screen. We therefore introduce the following approach.

We first use a large Δt in order to show the approximate result quickly. If the simulation ends without any changes of the object states such as position or shape, the Δt is set to half its previous size, then the simulation is executed again. This makes the result more accurate. If, however, any of the object states are changed, the simulation is re-started with a twice Δt. This realizes quick response of the simulation in exchange for accuracy. Since this Δt adjustment is performed automatically by the system, the user does not need to suggest an interval (Fig. 4).

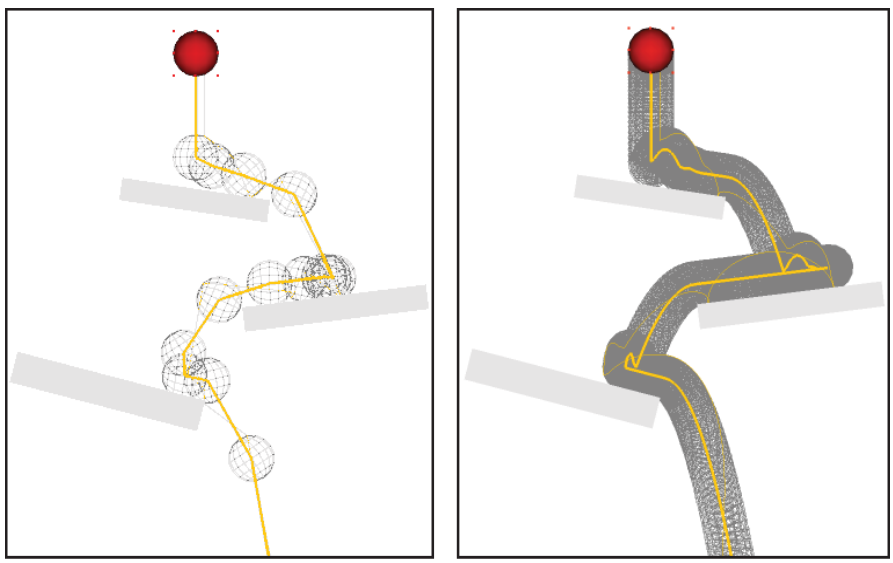


Fig. 4: The interval of the simulation is changed dynamically and automatically. The Δt of the left and right figures are 100msec and 5msec respectably. The simulated period is 10 sec.

3.2 Visualization of the Simulation Results

As a result of the physical simulation, the system obtains matrices containing information of the orientations and positions of the objects. By iterating the simulation at a specified time step, the motion of an object can be expressed as an array of matrices. Now, the problem is how to show the motion to the user.

If the system displays wireframe model drawings at every time step, as shown in Figure 5(a), it is sometimes difficult to visualize behavior because the screen becomes too cluttered, especially when numerous objects exist in a single scene or when the number of iterations is large. To avoid this problem, the system only draws objects located at positions where the direction of the motion has changed.

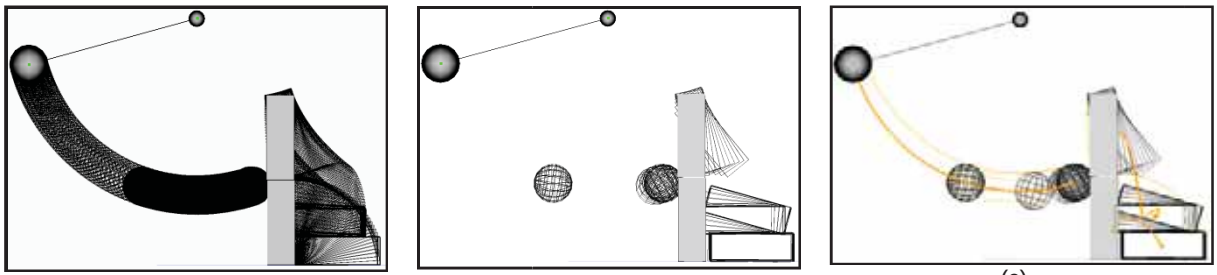


Fig. 5: The differences in the drawing that a pendulum smashing into a stack of two bricks: (a) drawing the object at all positions computed by the simulator, (b) drawing the positions where the direction of the motion is changed, (c) drawing the trajectory of the center and the most-moving vertex. Since the ball rotates after the smashing, two curves above and below the trajectory of the center are shown.

Specifically, we identify the position where the sign of d_t in the following equations becomes negative.

$$\mathbf{v}_t = \mathbf{M}_t\mathbf{p} - \mathbf{M}_{t-1}\mathbf{p}$$
$$\mathbf{a}_t = \mathbf{v}_t - \mathbf{v}_{t-1}$$
$$d_t = \mathbf{a}_t \cdot \mathbf{a}_{t-1}$$

Where $\mathbf{M}_t$ is an rigid-body transformation matrix of the object at time t, which is computed by the simulator, and $\mathbf{p} = (x, y, z)$ is the centroid, i.e., mean position of all the vertices of the object. Figure 5(b) shows the result of adopting this approach. We can see that objects are displayed only at the positions where the direction of the motions has been changed.

To clarify the behaviors of objects, we also display two loci for each object. One is the locus of the center of the object and the other is the vertex which has the longest incremental motion in the last time step. To find the longest incremental motion, the system traces all loci of vertices during simulation. We can see that the locus of the center shows the locus of the motion, and the locus of the vertex shows the rotational behavior. Figure 5(c) shows the result of this feature. Furthermore, the positions where collisions occur are visualized as red points, as shown in Figure 6. These features help users to more easily comprehend the motions of objects in a single image.

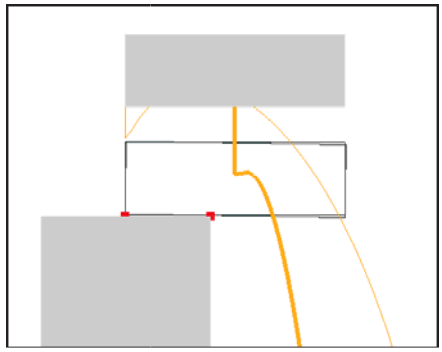


Fig. 6: A box falling onto the edge of an obstacle (Collision points are expressed as red dots).

3.3 Visualization of Stationary Objects

The visualization mentioned in the previous subsection works well for designing objects in which the trajectories have meaning. However, motion trajectory itself is not so important when we design a balance toy because our concern is whether the objects are stable. Accordingly, we added an option that can be used to switch the display of the object's motion. In the newly added display mode, the locus of the center of the object and the locus of the longest vertex are first calculated for each object. If the center is almost stable, the object is considered to be rotating. Then, two magenta arrows are displayed to indicate the rotation direction, as shown on the left in Figure 7. If the two loci are almost the same, the object is considered to be moving without rotation. If that occurs, a blue arrow that shows the movement of the object is displayed, as shown on the right in Figure 7. In other cases, we use two yellow arrows that show loci as mentioned in section 3.2.

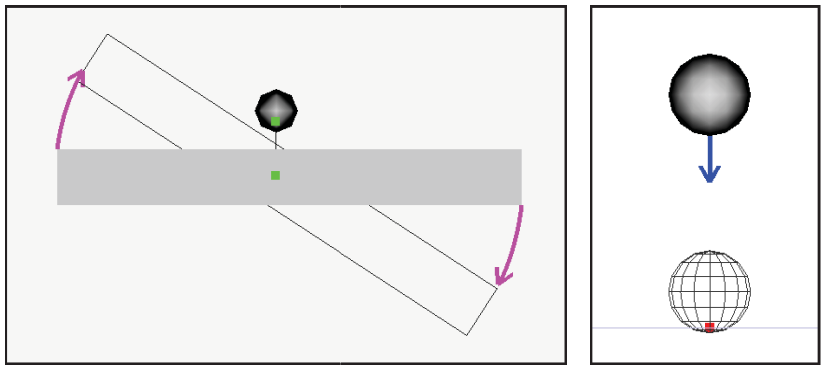


Fig. 7: Representation of rotation (left) and parallel translation (right).

4 SYSTEM EVALUATION

We asked 6 test users to carry out two different tasks with and without the proposed interface. With the proposed interface, the physical simulation runs in the background and the result is continuously shown to the user as a static visualization during editing. Without the proposed interface, the user explicitly runs a simulation by pressing a button and the result is shown as an animation. Therefore, test user completed two tasks. All users were university students who were familiar with computers but who had no experiences in designing 3D models with CAD.

Before executing a task, they were instructed on how to use our system and were given practice time and a practice task. The practice task was to adjust the position of a small pink sphere and a string to make a mobile sculpture stable (Fig. 8).

A test covering the same task with a different interface was given on a different day. And the test environment was as follows:

Model:	Dell Precision T5400
CPU:	Intel Xeon E5430 2.66 GHz (FSB: 1333 MHz)
RAM:	4 GB FB-DIMM with ECC 667 MHz
VGA:	NVIDIA Quadro FX 4600 (VRAM: 768 MB)
Misc:	Microsoft Vista sp1, Java 1.6.0_16 + JOGL 1.1.1

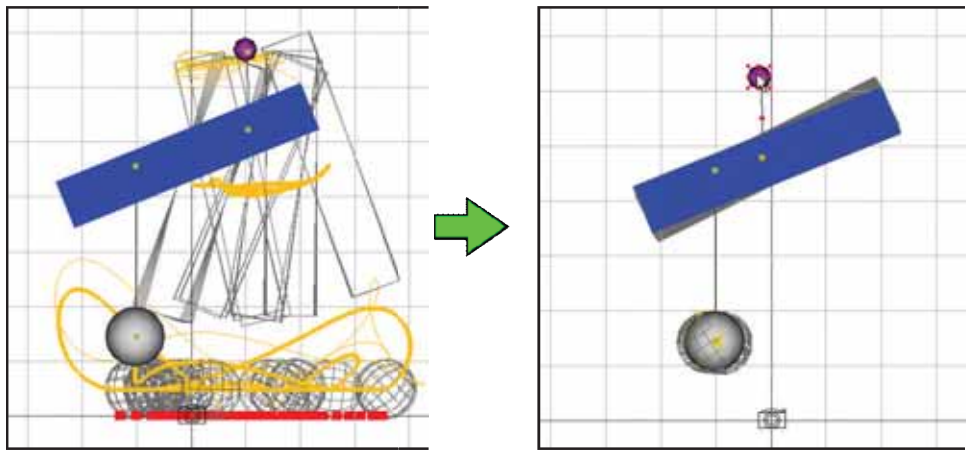


Fig. 8: Practicing to use our system.

4.1 Test Task 1: Adjusting the Starting Position of a Dropping Sphere.

The first test task was to find an appropriate initial position of a sphere where it touches three boxes while dropping down to the floor, as shown in Figure 9. The left of Figure 9 shows the default condition and the right shows one of the correct answers. The display mode was set to “motions with loci”.

This task is regarded as the process of designing bagatelles (toys that involve the rolling of balls) or Corinth games (games in which objects collide) [4]

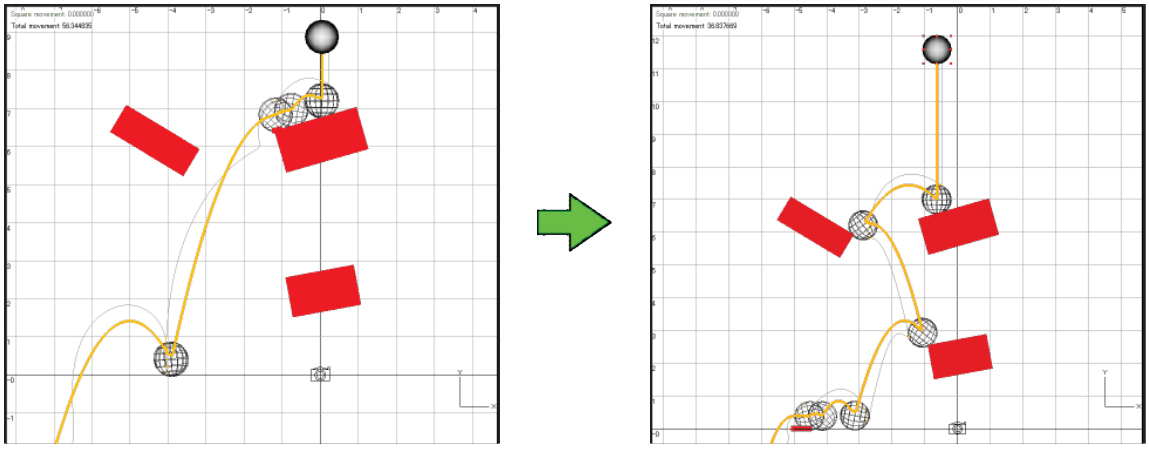


Fig. 9: Test Task 1: Adjusting the starting position of a dropping sphere.

Figure 10 shows the time needed to complete the task. Tester A, B, and C first performed a task with the proposed interface. On the other hand, tester D, E, and F first performed a task without the proposed interface. As shown, the operation time is reduced to almost half on average with our system. Some testers said, “I could easily guess how to find the right answer while seeing the result of the simulation,” and “I could do it efficiently because the system automatically showed the result of the simulation.” Almost all testers found an answer smoothly. Since user B found the almost correct answer by chance in a very short time, even without physical simulation, the time of user B is short overall compared with other users.

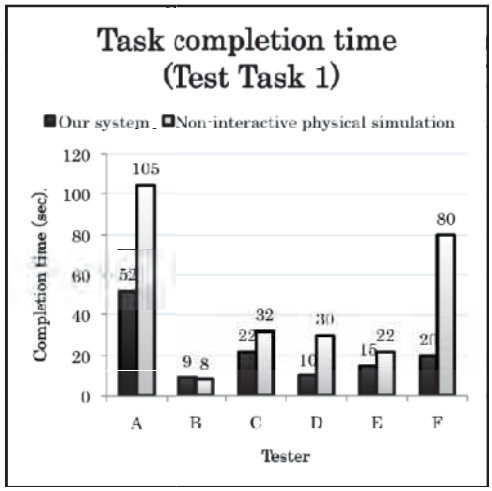


Fig. 10: Graph of the completion times of Task 1.

4.2 Test Task 2: Adjusting the String Suspension Points in a Mobile.

The second test task was to adjust the positions of strings and weights of a mobile sculpture in order for it to be stably balanced, as shown in Figure 11. The left of Figure 11 shows the default condition and the right shows one of the correct answers. The display mode was set to “motions with arrows”. The completeness of the task was determined by measuring the average movements at the top left of the display shown as a light blue text. We assumed the task was complete when the average value

became less than 1 mm/sec. The test users were allowed to move the positions of the strings, the gray spherical weights and the red spherical fulcrum.

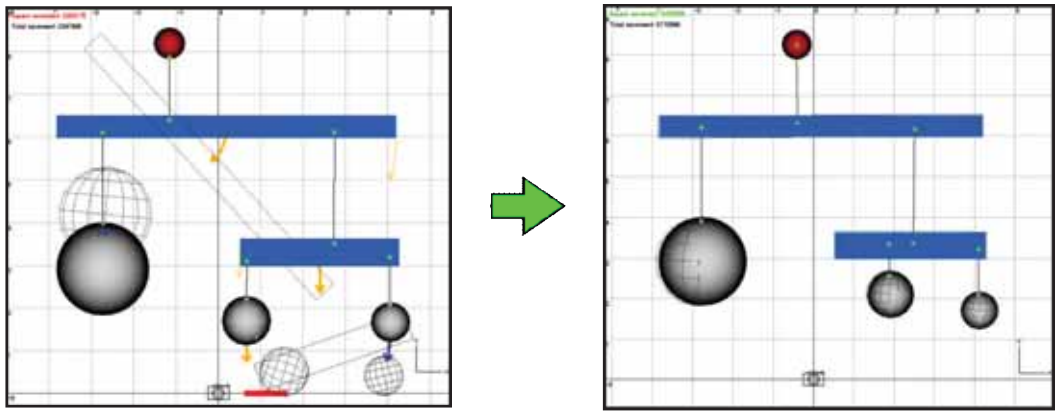


Fig. 11: Test Task 2: Adjusting the string suspension points in a mobile.

Figure 12 shows the time needed to complete the second task. Tester A, B, and C first performed a task with the proposed interface. On the other hand, tester D, E, and F first performed a task without the proposed interface. As shown, the testers could reduce the task completion time to less than one third (on average) with our system. And we obtained the comment that our interface made it “easy to see and understand the behavior.”

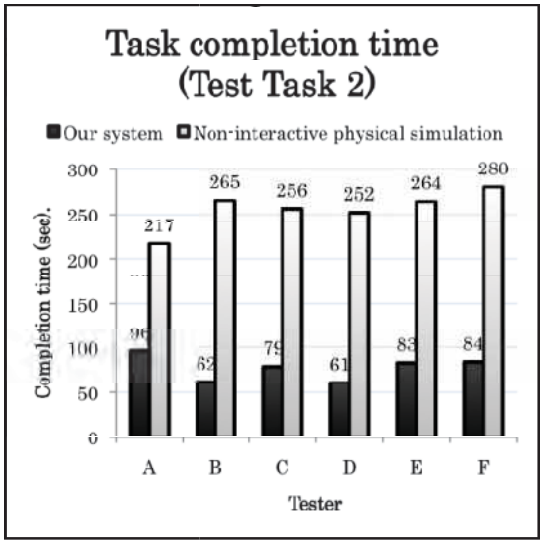


Fig. 12: Graph of the completion times of Task 2.

One test user moved the central sphere to the right side, as shown in Figure 13. The result was balanced, but we did not anticipate this answer. This shows that our system can help find several unanticipated alternatives to a particular solution. Even though some optimization approaches could help users find valid solutions quickly, our system enables users to discover new alternatives.

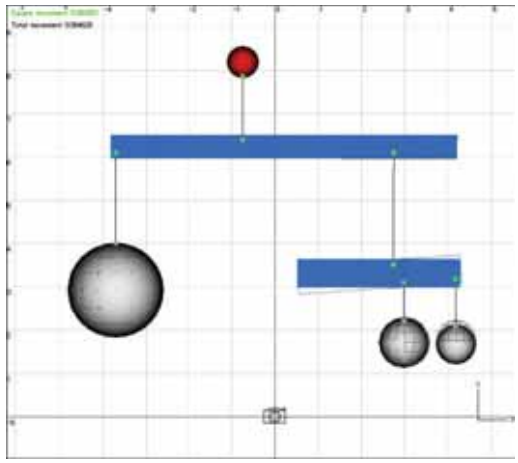


Fig. 13: Another solution to test task 2.

5 DESIGN RESULTS

The authors of this system designed and created several balance toys with our system. Figure 14 provides an example. It was difficult to balance this mobile because the user must not only design “the shape of kinetic objects” but also their suspension and/or contact points, and it incorporates a loop of connected parts. However, we completed the design in approximately 15 minutes, after which we used a laser cutter to fabricate components for an actual model out of 5 mm acrylic resin board. Figure 15 shows another example. This one was also difficult to design without our system because it has a 3D configuration.

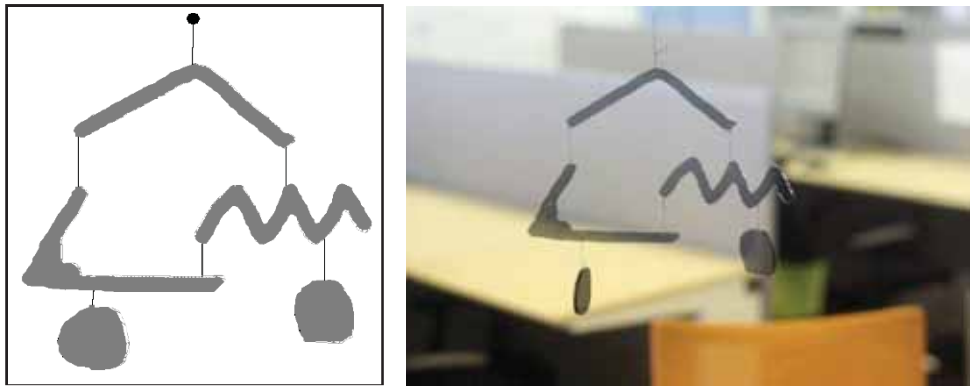


Fig. 14: Example of a mobile.

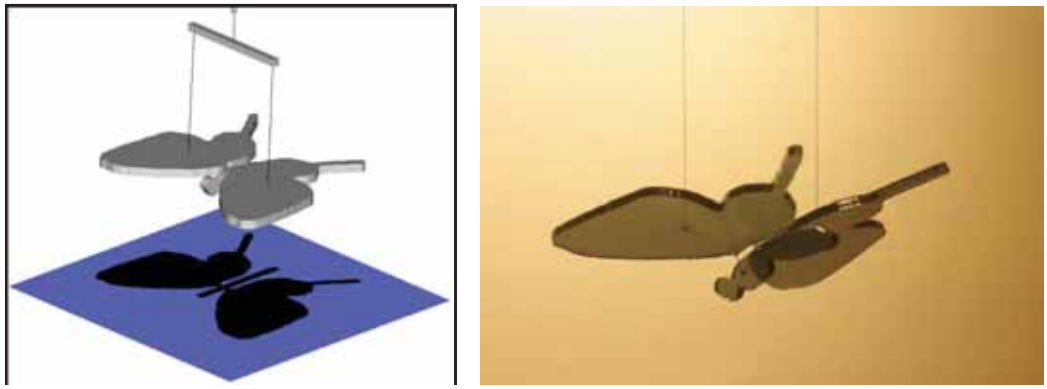


Fig. 15: A second example of a mobile.

Figure 16 shows an example designed by a test user. We provided her with approximately five minutes of instruction at the beginning of the session. After that, we did not put any limits on her design time. We also provided guidance on the use of the software, as required, during her design work. It took approximately one hour to complete her design. She reported that she successfully created the shape she desired. She also identified some limitations of the current system, such as the lack of an undo function and the need for a way to add holes to some parts.

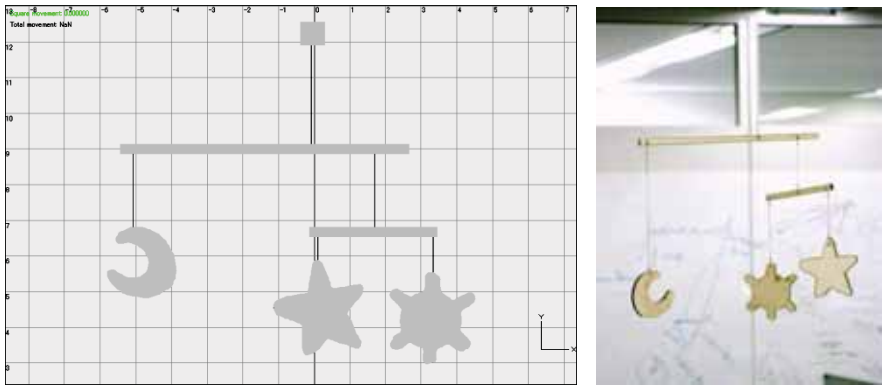


Fig. 16: Example of a mobile created by a test user.

6 CONCLUSIONS AND FUTURE WORK

We propose a system that integrates geometric modeling and physical simulation in order to design the shapes of kinetic objects. We have presented an implementation technique that automatically adjusts the time step for simulations, based on user operations. The results of a simulation are visualized in the modeling window. After we had implemented the system, we demonstrated that we could design kinetic objects with our system more efficiently than could be accomplished with other current systems. We also created actual mobile sculptures using our system and observed that they were well balanced.

Although our system worked well for designing simple kinetic objects, some limitations were identified. Our system makes it easy to understand the motions of objects by displaying wireframes and colored arrows, but the visualization becomes too complex when the number of objects on the screen increases substantially. We plan to add a function that enables the user to select target objects

to show their motion. Then we would like to try to design more complex kinetic systems that involve a lot of collisions and complicated motions.

Because an object that is almost unstable is more fun than one that is perfectly stable, the user may wish to intentionally design objects that move widely due to minor influences while remaining "balanced" in the long run. However, currently there is no way to tell the difference between these types of objects in our system. Supporting the design of balanced but almost unstable objects will be an interesting subject of future work. Slow oscillation frequencies might be a good criterion to indicate the near-instabilities of balancing toys.

Integration of the modeling interface and of the simulation engine is expected to be useful for more general systems. In the future, we plan to test various simulation methods, such as fluid simulation and finite element methods, which support a greater variety of objects. The key challenges are increasing the speed of the simulation and presenting the simulation results in a way that does not overwhelm the user. In summary, the methods presented in this paper can be useful in many domains.

REFERENCES

[1] Frank, B.: Kineticus, http://www.kineticus.com/.

[2] Alvarado, C.; Davis, R.: Resolving ambiguities to create a natural computer-based sketching environment, In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, Washington, 2001, 1365-1371.

[3] Goldman, D. B.; Curless, B.; Salesin, D.; Seitz, S. M.: Schematic storyboarding for video visualization and editing, ACM Trans. Graph., 25(3), 2006, 862–871.

[4] Reed, D.: A little about the machines; Faculty.ccp.edu, http://faculty.ccp.edu/faculty/dreed/Campingart/pachinko/about.htm.

[5] Ernerfeldt, E.: Phun, http://www.phunland.com/wiki/Home.

[6] DarthGak: sketchyphysics - Physics plugin for Sketchup, http://code.google.com/p/sketchyphysics/.

[7] Google Inc.: SketchUp, http://sketchup.google.com/.

[8] Popovic, J.; Seitz, S. M.; Erdmann, M.; Popovic, Z.; Witkin, A.: Interactive manipulation of rigid body simulations, In SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, 209–217.

[9] Mitani, J.; Suzuki, H.: Making papercraft toys from meshes using strip-based approximate unfolding, ACM Trans. Graph., 23(3), 2004, 259–263.

[10] Nienhaus, M.; Dollner, J.: Dynamic Glyphs – Depicting Dynamics in Images of 3D Scenes, In Proceedings of Third International Symposium on Smart Graphics, 2003, 102–111.

[11] Straub, M.; Needham, J.: Digitally assisted billiards. http://hackaday.com/2008/12/10/digitally-assisted-billiards/.

[12] Umetani, N.; Development of integrated design analysis software with interactive UI, In Proceedings of the 50th Symposium of IPS Japan, 2009, 105-110.

[13] NVIDIA corp.: Physx 2.8.1, http://developer.nvidia.com/ob ject/physx.html.

[14] Ole Flensted: Bauhaus mobile, Flowing rhythm, http://www.flensted-mobiles.com/images/thumbnls/thumbs0.htm.

[15] Eastman, P.: Art of illusion, http://www.artofillusion.org/.

[16] Prometech Software: PhysiCafe, http://www.prometech.co.jp/products/physicafe.html.

[17] Tang, S. H.; Linton, M. A.: Pacers: Time-elastic Objects, ACM Symposium on User Interface Software and Technology, 1993, 35-43.

[18] Weyrich, T.; Deng, J.; Barnes, C.; Rusinkiewicz, S.; Finkelstein, A.: Digital bas-relief from 3D scenes, ACM Transactions on Graphics (Proc. SIGGRAPH), 26(3), 2007.

[19] Mori, Y.; Igarashi, T.: Pillow: interactive pattern design for stuffed animals, In SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches, 74.