Objects Collision Model for Physics Teaching and Learning

1051963

Abstract

Sometimes students may find it hard to figure out the subsequent movement of two objects after a collision between them, because they can only learn the knowledge from their textbook which is not concrete enough for them to understand. This paper uses Empirical Modelling (EM) approach to develop a model of physical collision between two objects, simulating the collision process, to provide a concrete view of the process to physics students and help them to gain a better understanding of collision. As it is for the tutorial, only ideal and non-deformed objects are considered in the model.

1 Introduction

Collision is one of the most important but also complexed parts in Mechanics. When learning physics, beginners always have to face difficulties in fully understanding it, which also brings stress to teaching physics. This paper introduces a model based in EM concept to help students learn objects collision.

EM aims at teaching people useful things via observation and experience with models. The collision model in this paper provides another way to students to observe the movement of two colliding objects, by which they can learn the knowledge in an intuitive way, and which is actually more fun than only reading the textbook.

2 Physics Formulae

Basiclly, there are two formulae used in this model. One is for Perfectly Elastic Collision, and the other is for Completely Inelastic Collision. Non-perfect Elastic Collision is not considered in this model, because it needs more conditions to decide objects' movements and to calculate their attributes.

2.1 Perfectly Elastic Collision

In this type of collision, after two objects collide, their speed must be different with each other (contrary to Completely Inelastic Collision). According to Conservation of Momentum Law,

$$m_1v_1 + m_2v_2 = m_1v_1' + m_2v_2'$$

where m_i is the mass of $object_i$, v_i is speed of $object_i$ before the colliding moment, and v_i' is the speed of $object_i$ after the colliding moment.

As in Perfectly Elastic Collision, there is no energy loss caused by the collision, then according to Energy Conservation Law,

$$\frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 = \frac{1}{2}m_1v_1'^2 + \frac{1}{2}m_2v_2'^2$$

and combining with the Square Roots Formula,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

the speed v_i' can be then calculated.

2.1 Completely Inelastic Collision

In this type of collision, the two objects will gain the same speed after the collision and "stick" together in their subsequent movement. Then the Conservation of Momentum Law becomes,

$$m_1v_1 + m_2v_2 = (m_1 + m_2)v'$$

where v' is the speed of the two objects and can be calculated from this formula.

3 The Collision Model with EM Concepts

The collision model is an assistant to help students learn physical collision better. Its main function is to show the process of a collision of two objects and their subsequent movement. During running the model, it allows users to set attributes of objects and other necessary data by their own needs and preference. And they will get the feedback (values of important data) from the model in real time.

3.1 Attributes Setting

In the collision in physics, there are four things should be concerned, which are respectively the mass of the two objects, the initial speed of them, the initial distance between them, and the friction force they are taking.

In this model, they become the observables and can be set by users. See Figure 3-1 and Figure 3-2.

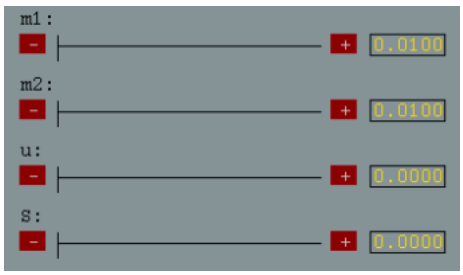

Figure 3-1: Initial Value of Some Observables

Figure 3-2: Setting Some Observables

As can be observed in Figure 3-1, the initial value, which is also the minimum value, of the friction coefficient and the distance between two objects are both zeros. However, the value of mass of the two objects are 0.01, one reason for that is that in real world there is no objects that weight zero. Another reason is for the safety of the model, as mass will be denominators in the calculation. Figure 3-2 shows the setting of the initial speed of the two objects where the sign of the value represents for the direction of object's movement.

To set the values, users can both use the stick bar and the buttons on its two sides, which is to make the values more accuracy.

3.2 Collision Type Choosing

As described above, there are two types of collision in this model, which will cause totally different influence on the process of the collision. Users are able to choose collision type by simply click the radio button that corresponds to that type. See Figure 3-3.

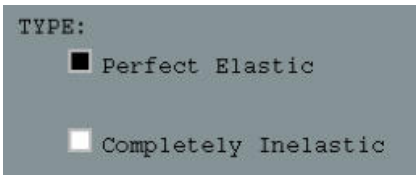

Figure 3-3: Radio Buttons

3.3 Main Window

The function of main window is to display the whole process of the collision. The collision detection is relatively simple: it detects the coordinate of the upper right point of the left object and the upper left point of the right object on the x-axis, once the two values are equal, it means that the two objects collide.

However, as the window is limited by its width and height, when the speed of objects is too high, objects will move out of the window which will not be able to observe any more. To solve this problem, an auto-zoom function is added to the window, which will automatically detect if objects are moving to the edge of the window and then decide whether zoom in or out or not zoom. Figure 3-4 shows the initial scene in the window, and Figure 3-5 shows the zoomed scene in the window.

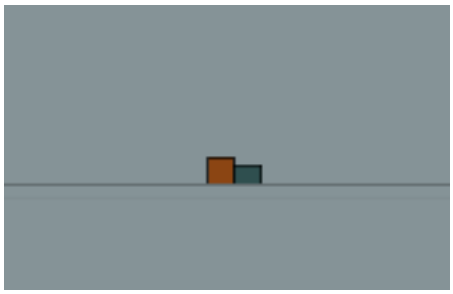

Figure 3-4: Initial Scene

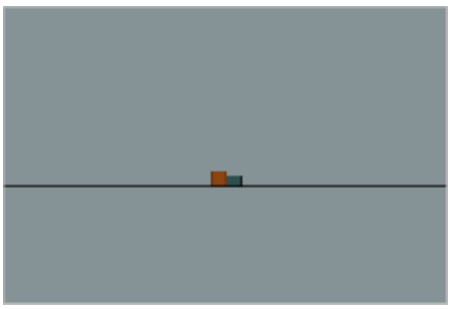

Figure 3-5: Zoomed Scene

To achieve the zoom function and the animation of movements, the *edenclock* is used to update the value of object's position and the zoom variable everytime the clock variable changes. Besides, the speed values and distance between two objects are also updated and displayed to the user, which will be introduced in the following section.

3.4 Feedback to Users

As a tutorial tool, it is of great importance to let users get enough data, from which they can learn some details in the process of collision and eventually understand the collision better.

In this model, user are able to get real-time speed of objects, the distance between them, and also their instantaneous velocity at the colliding moment and the loss of energy in the collision. These observables are divided into two groups. One is the Current Values group, which will display the value of relative observables in real-time; the other one is the Colliding Moment group, which is used to show relative data right after the collision. The former group is much more complexed than the later group, because all the data listed there needs be updated during the movement of the objects. The *edenclock* is also needed in this action. More exactly, everytime the clock argument changes, the observables in the group get updated. Figure 3-6 shows the values of observables after a completely inelastic collision.

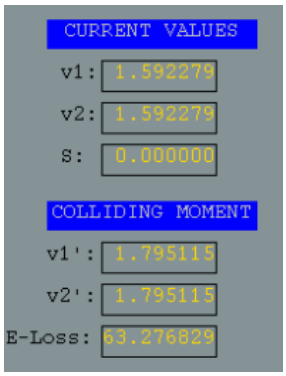

Figure 3-6: Feedbacks to Users

3.5 Dependancies in The Model

Dependancy means that the change on one observable will lead to the a change on another observable. It is quite useful in building a model. For example, if *Y* is an observable to be displayed on the screen, which satisfies the condition *F(X)*, where *F* is a function and *X* is another observable and is changed by the user. Hence, everytime the user changes the value of *X*, s/he will get a new value of *Y* that s/he might be concerned about as feedback.

In the model, there are numerous dependancies between observables. For instance, to implement the window zoom function, a factor is needed to make the change on width proportional to that on height, or the scene will be distorted. Once the factor changes, the zoom on *x* direction and *y* direction will both have a change. In addition, the value of the factor does not change manually, it detects if one of the objects is going to move out of the window. If it is, the factor will continue enlarging itself to keep the object in the window. Thus, there is also a dependancy between the factor and the position of objects. Some relative codes are showed below as examples.

```
x_zoom is 3 * factor;
y_zoom is 2 * factor;
iSpeed_input_str is str(iSpeed_input);
pSpeed_input_str is str(pSpeed_input);
acceleration is fric_input * 9.8 / 50 /10;
```

4 Other Attempts

To make it clearer to show the zooming of the window, a narrow horizontal bar was added into the main window which was used to represent the initial distance. It was supposed to remain constant after the two objects started to move, so when the window zoomed, it would zoom with the window, from which the zooming function could be represented better. However, the result after adding this bar was not as good as expected. Because it just looked like another object in the scene, and it did not make the zooming function better represented. Thus, the idea is not used in the model at last.

5 Evaluation

Generally speaking, the model has a good and real performance, which reaches the goal of being an educational assistant. Although it looks simple, it contains nearly all the key elements in the collision in physics, and it is easy to use as the user interface is very intuitive.

However, there are also some drawbacks existing. For example, when the friction coefficient is set to be zero, objects will keep moving in a constant speed without stop. So there will be a moment when they move out of the platform. Additionally, with the enlarging of the distance between the two objects, the main window will continue zooming out, which will make the objects and the platform become too small to be observed. Another flaw is that when the object's speed is too

high, they still might move out of the window even if there exists the auto-zoom function, because of the limitation of *edenclock* which cannot update the clock argument that fast.

6 Conclusion

It has been showed in the model that EM has great ability in modelling, and also has the potential in education area to make it more interesting and more efficient.

Furthermore, as it is not hard to develop a simple tool, it might attract more people without professional knowledge in Computer Science to use it building models that are useful to them.

Acknowledgements

The author would like to thanks Dr. Meurig Beynon for his constructed suggestions on the idea of the model.

References

S. Yung. (1989). *Definitive notation for sCreen LayOUT (GUI window layout)* [Online]. (URL http://www2.warwick.ac.uk/fac/sci/dcs/researcr/em/notations/scout/). (Accessed 10 Jan 2011).

E. Yung, *et al*. (2005). *EDEN Handbook* [Online]. (URL http://www2.warwick.ac.uk/fac/sci/dcs/research/em/software/eden/langref/). (Accessed 10 Jan 2011).

A. Ward. (Version 1.66). *Eden Quick Reference*.

Y. Yao. (2009) *Sniper*.

M. Beynon. (2008). *Revised JUGS model*.

A. Ward. (1997). *Cat Flap*.