1. Sergey VYATKIN^1, 2. Tetiana KOROBEINIKOVA^2, 3. Pavlo MYKHAYLOV^3, 4. Roman CHEKHMESTRUK^4, 5. Oksana VODZINSKA^5, 6. Vasyl OVCHARUK^6, 7. Andrzej KOTYRA^7, 8. Gulzhan KASHAGANOVA^{8,9}, 9. Zhazira JULAYEVA^{10}

Institute of Automation and Electrometry(1), Lviv Polytechnic National University Ukraine (2), 3D GNERATION GmbH, Germany (3), 3D GENERATION UA, Vinnitsa, Ukraine (4), Kyiv National University of Technologies and Design, Ukraine (5), Vinnytsia National Technical University (6), Lublin University of Technology (7), Institute of Information and Computational Technologies CS MES RK (8), Kazakh-American University (9), Academy of Logistics and Transport (10)
ORCID: 1. 0000-0002-1591-3588; 2. 0000-0003-2487-8742; 3. 0000-0001-5861-5970; 4. 0000-0002-5362-8796, 5. 0000-0002-1246-7156; 6. 0000-0002-8443-5460; 7. 0000-0002-9442-6090; 8. 0000-0001-8150-1621

doi:10.15199/48.2021.09.07

A method for calculating trajectories independent of the explicit determination of an object's equation of motion

***Abstract**. A method for calculating the trajectory of a physical object that does not rely on explicitly setting the equation of motion is proposed. We analyze the problems and formulate requirements for the developed model. The choice of the motion algorithm is justified. The paper describes the technical part of the developed model and the logical structure of the program. The application contains the results of testing the system*

***Streszczenie.** Zaproponowano metodę obliczania trajektorii obiektu fizycznego, która nie polega na jawnym wyznaczaniu równania ruchu. Przeanalizowano problemy i sformułowano wymagania dla opracowanego modelu. Uzasadniony jest wybór algorytmu ruchu. W pracy opisano część techniczną opracowanego modelu oraz strukturę logiczną programu. Aplikacja zawiera wyniki testowania systemu. (**Metoda obliczania trajektorii niezależnie od jednoznacznego wyznaczania równania ruchu obiektu**).*

Keywords: animation, motion model, Lagrangian mechanics, function variation, inertia tensor.
Słowa kluczowe: animacja, model ruchu, mechanika Lagrange'a, zmienność funkcji, tensor bezwładności.

Introduction

Animation is important in creating simulators, video games, as well as in modeling physical phenomena and experiments [1-4]. Virtual objects created by the designer are placed in the virtual scene, which should move, collide and rotate in the closest way to reality. Currently, there are many motion models used in simulators and training programs. The main problem with using existing models is that they rely on empirically created engineering equations of motion. The extreme complexity of creating equations does not allow optimal use of the capabilities of such systems. In this situation, it is desirable to create a universal model [5-7].

Let's consider the existing methods of motion modeling. The movement of an object is defined by actual physical equations that are explicitly defined. However, the differential equations themselves are difficult to calculate in real time. This method is usually used to implement the behavior of objects in experiments where speed is not important, but only the result itself is important [8-10].

The movement of an object is modeled by substituting empirically selected constants into the equation of movement of the object to create a more realistic behavior of the object (i.e. using simple formulas, to calculate the position of the object, we obtain the required behavior by selecting constants).

The simplest and most frequently used method that does not require physical calculations is to set the trajectory of the body. Track-the trajectory is set by the designer and remains unchanged [10-12].

The main way to build a model of an object's motion is to use its technical characteristics to calculate the equation of motion. In this equation, empirically selected constants are substituted for more plausible behavior of the object. In this case, "fuzzy logic" methods are often used. This approach rigidly binds the developed model to the properties of the object and its environment. In this situation, it is desirable to create a universal model of movement. Such a generalized model can be a motion algorithm based on Lagrangian mechanics [12].

Let's combine the above methods in Table 1.

The purpose of this work is to create a generalized model of motion that is independent of the explicit setting of the object's motion equation [10,12]. The following tasks must be completed:

- Analysis of existing models.
- Formulation of requirements for the developed model.
- Development of the theoretical base of the model.
- Realization of algorithm.
- Testing and debugging the system.

Table 1. Comparison of systems using Lagrangian mechanics with conventional models of motion

Sensor type	Lagrangian mechanics	Other models
Rigidly binds the motion model to the properties of the modeling object based on other algorithms	**–**	**+**
Is a universal model (suitable for any objects)	**+**	**–**
Takes into account object properties and properties its surroundings	**+**	**+**
Explicitly uses the equation of motion of the object (built on the explicit definition of the equation of motion)	**–**	**+**

Theoretical basis of the model

A solid body can be defined in mechanics as a system of material points whose distances are constant (Fig. 1).

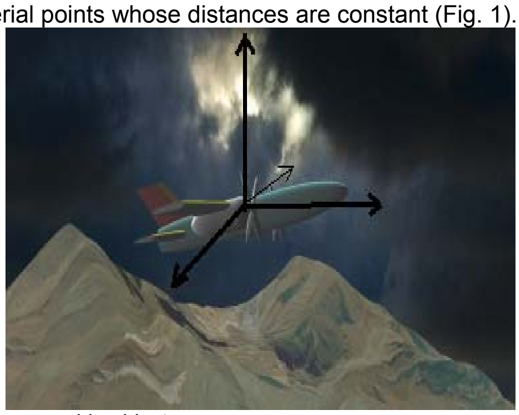
Fig.1. The movable object

Systems that actually exist in nature can, of course, satisfy this condition only approximately. We will consider a solid body as a discrete set of material points [5].

Each mechanical system is characterized by a specific function $L(q_1,...,q_s,\dot{q}_1,...,\dot{q}_s,t)$ or $L(q,\dot{q},t)$ in a short entry.

The principle of least action

The most general formulation of the law of motion of mechanical systems is given by the so-called principle of least action (or Hamilton's principle). Let the system occupy certain positions at moments of time $t=t_1$ and $t=t_2$, characterized by two sets of values q_1 and q_2. Then between these positions the system moves in such a way that the integral

(1) $$S=\int_{t1}^{t2}\mathrm{L}(q,\dot{q},t)dt\to\min$$

had the smallest possible value (true for each of the relatively small sections of the trajectory). The function L is called the Lagrange function of this system, and the integral is called the action.

Method of function variation

Let $q=q(t)$ be just the function for which S has a minimum. This means that S increases when replacing $q(t)$ with any function of the form

(2) $$q(t)+\delta q(t),$$

where $\delta q(t)$ is a function small in the entire time interval from t_1 to t_2. (it is called a variation of the function $q(t)$).

Since for $t=t_1$ and $t=t_2$, all the compared functions (2) must take the same values q_1 and q_2, then it must be:

(3) $$\delta q(t_1)=\delta q(t_2)=0.$$

The change in S when replacing q with $q+\delta q$ is given by the difference

(4) $$\int_{t1}^{t2}L(q+\delta q,\dot{q}+\delta\dot{q},t)dt-\int_{t1}^{t2}L(q,\dot{q},t)dt$$

thus, the principle of least action can be written as

(5) $$\delta S=\delta\int_{t1}^{t2}L(q,\dot{q},t)dt=0,$$

or by performing integration:

(6) $$\int_{t1}^{t2}\left(\frac{\partial L}{\partial q}\delta q+\frac{\partial L}{\partial\dot{q}}\delta\dot{q}\right)dt=0$$

(7) $$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{q}_j}\right)=\frac{\partial L}{\partial q_j}$$

where $L=T-U$ is the lagrangian of the system.

These are the desired differential equations; they are called Lagrange equations in mechanics. If the Lagrange function of a given mechanical system is known, then the Lagrange equations establish a relationship between accelerations, velocities, and coordinates, i.e. they represent the equations of motion of the system.

(8) $$\frac{\partial L}{\partial\dot{q}_j}=p_j$$

Requirements for the new model

Dependence on the degree of linearity of movement. Conditions for the applicability of non-relativistic classical mechanics:

(9) $$\dot{q}<<c,\ \hbar<<L$$

The necessary parameters must be set when creating the object

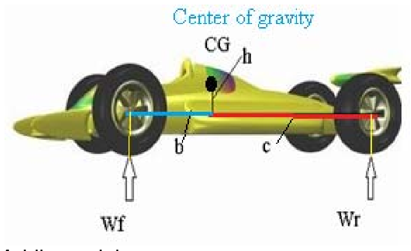

Fig. 2. Mobile model

Realization of algorithm

The principle of least action

$$S=\int_{t_1}^{t_2}Ldt\to\min$$

The ends of the trajectory are fixed $\delta_{gr}=0$.

There is no explicit equation of motion, so based on the principle of least action, it is necessary to find the next position of the body using the method of variation. To do this, we will make a selection from the array of Lagrangian values by the smallest value. Then, compare the value of the Lagrangian and the point where it was calculated. Next, we find the final coordinates of the object's position using the Euler method.

(10) $$\begin{aligned}\delta S&=\int_{t1}^{t2}\delta Ldt=\int_{t_1}^{t_2}\left(\frac{\partial L}{\partial\dot{q}}\delta\dot{q}+\frac{\partial L}{\partial q}\delta q\right)dt=\\&=\int_{t_1}^{t_2}\left(\frac{\partial L}{\partial q}-\frac{d}{dt}\frac{\partial L}{\partial\dot{q}}\right)\delta qdt=0\\&\forall\delta q\Rightarrow\frac{\partial L}{\partial q}=\frac{d}{dt}\frac{\partial L}{\partial\dot{q}}\end{aligned}$$

11) $$T=\sum_i\frac{m_i}{2}(\dot{\vec{r}}_i+[\vec{\Omega}\times\vec{r}_i])^2\qquad U=-\sum_i m_i\vec{g}\vec{r}_i$$

Using the variation method, a cube of possible positions is constructed around the object. At each position, the Lagrangian is recalculated and entered in the array of correspondences along with the coordinates of the object. Then the action value is calculated and its minimum is found. Going back by the key (the Lagrangian value) in the array, we find the position of the object. The algorithm works cyclically (by frames). Every frame is subdivided into pieces (Fig. 3).

The main requirement for this model is its abstraction from the specific properties of the object: that is, when creating objects, the designer sets a list of necessary parameters (such as the inertia tensor, center of gravity, initial coordinates and speeds), and the model calculates the system of objects by changing their trajectories [13].

The model also depends on the degree of linearity of motion, i.e. in the case of nonlinear motion; the collision detection algorithm cannot accurately process collisions, so the parameter of approximation to linearity is introduced [14]. Depending on this parameter, a time interval is selected where the movement can be considered linear and where collisions are processed.

The same parameter is the unit of measurement into which the cube of possible object positions is divided. This allows the collision detection module to make sure that no collisions are missed

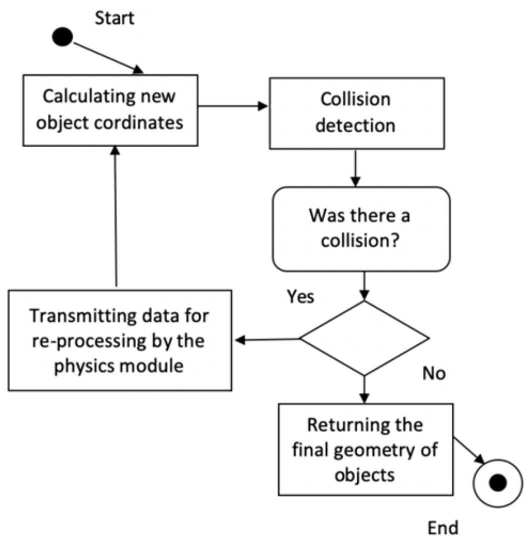
Start
Calculating new object cordinates
Collision detection
Was there a collision?
Yes
Transmitting data for re-processing by the physics module
No
Returning the final geometry of objects
End

Fig. 3. Proposed algorithm

Computation of reaction to collision is performed after determining pair of objects VO_pVO_q collided at the earliest time. To make this computation independent of particular application requirements we use technique of message handling callbacks well known in object-oriented programming. Callback functions encapsulate application area specific routines for computation of reaction to collision. In implementation of our algorithm, several sample callback functions are supplied. Messages sent to virtual objects contain parameters of collision and are handled by callback functions. Messages of different type (class) are discriminated using 'extension type fitness' mechanism analogous to that of Oberon language rather than C++ run-time type information mechanism.

So, when collision is detected a message with collision parameters is sent to collision participants. Each virtual object is assigned a priority to handle messages. The message is sent first to the object with higher priority. If priorities are equal then choose of object to send message first to be random. The message to the second object is sent (or not sent) by the first object. This is done because the contents of the message can be changed after computing reaction to collision for the first object. An object received the massage discriminates the message type and calls corresponding callback function with two parameters, the object itself and the message. Callback function computes reaction to collision and changes dynamic parameters of the caller object and, if it is the first object to handle the message, contents of message may be changed. Along with changing dynamic parameters of the object the callback itself can be replaced during handling the message, providing by this a different way to handle future collisions as well as different behavior of the object in respect to other events (messages).

This technique of message handling callbacks being very versatile also spends minimal computational resources. Overhead comprising time to discriminate message type and call function by pointer is hardly noticeable as compared with computations on wide and narrow phases of collision detection.

The weakest place of the proposed algorithm is possibility of (almost) infinite loop in case when number of detected collisions for fixed time interval is too big. Since quantity of objects in virtual environment is finite, so at least one object can be involved in an infinite number of collisions. Discrete representation of real numbers in modern computers gives rise to the fact that the infinite number of collisions for such object will occurs at the same time τ_∞ in the considered time interval. Otherwise current time interval would decrease on each step of the algorithm and inevitably would reach 0 due to discrete representation of real numbers but in this case, infinite number of collisions would happen at $\tau_\infty = T$.

Therefore, we see that infinite loops in our algorithm could take place in the situation when one object begins endlessly to collide with other objects in virtual environment. The situation so explicitly formulated can now be detected easily enough and therefore dead loop can be broken. This break improves the proposed algorithm robustness but as a rule, it means that computation accuracy is not enough or there are errors in modeling virtual environment (wrong choice of collision parameters, wrong change of dynamic parameters of objects etc.) since in actual environment no object can collide infinitely with others in finite time interval. In our algorithm, number of collisions for each object is counted and when it is greater some threshold the corresponding object does not take place in future computations on this step.

Setting initial conditions

The algorithm requires the following characteristics to calculate the position of an object:

1. Inertia tensor.
2. Mass.
3. The center of gravity.
4. The initial coordinates.
5. The speed of both translational and rotational.

The environment for creating a virtual environment provides a standard task. In this case, the speeds and coordinates are initialized with zeroes, and the center of mass will be in the reference center of the local coordinate system of the object.

Application of the method for medical tasks

If the Lagrange function of a mechanical system is known, then the Lagrange equations establish a relationship between accelerations, velocities, and coordinates, i.e. they represent the equations of motion of the system (equation 8).

Gait can be analyzed in terms of energy exchange, using the Lagrange method, and inverse kinematics. This will allow you to get models of both normal and prosthetic gaits. Designing and testing a prosthesis using computational tools is very important, and prototyping allows you to confirm the functionality of the prosthesis and the behavior of its components and materials. This allows you to set the maximum range of movements and deformations of the device before its final assembly. You can also evaluate the behavior of an element in a functional environment. Understanding the response of the biomechanical connection between the prosthesis and the patient provides developers with tools to understand and adapt the effects of the prosthesis in relation to gait. The design and adaptation of the prosthesis requires knowledge of the mechanical behavior of both its parts and the integrated final design, since it responds to the mechanical loads that it will bear under the conditions of use. The advantages of using computing tools in medicine and engineering are well known. In medicine, acquiring clinical skills before contacting patients is appropriate because it provides a controlled and safe environment, as well as establishing specific situations or scenarios as needed. The method consists in obtaining a mathematical model of the prosthesis and creating a simulation. Decoded information with the preservation of the physical and geometric characteristics.

The mechanical model is imported, preserving such characteristics of the prosthesis as mass, inertia, and coordinate system. The representation of the prosthesis is implemented in the form of a block diagram that allows you to relate it to the gait model.

The structural diagram of the prosthesis contains a world coordinate system, providing support for joints and moving parts, and "body" elements representing solid bodies that have physical variables such as moments of inertia, mass, and acceleration. The "body" has a center of gravity and points of mechanical connection with other parts. There are rigid bodies, such as the bushing and some connecting elements.

Conclusion

Search for the smallest key value in the <set> container (standard STL type). The key is the value of the Lagrangian at each point of the area adjacent to the object's position coordinate. Its calculation and introduction to <set> occurs with the consideration of forces acting on the object (potential forces are included in U, vortex forces are considered by the algorithm as additional pulses). Finding the final coordinates of the object's position using Euler's method.

A method for calculating the trajectory of a physical object is developed that does not rely on an explicit setting of the equation of motion, and a software module is implemented on its basis. The created software module can be used in modern simulators. Since the method does not require an explicit equation of motion, it can be used to calculate complex systems.

We have designed and implemented the algorithm for modeling interaction of physical objects in virtual environments in real time. This algorithm detects collision occurrence, determines its spatial and temporal localization with specified accuracy and computes reactions of objects for collision. Advantages of the algorithm

1. The proposed algorithm implements dynamic approach to collision detection and processing in real-time as opposed to discrete approach proposed in most published papers and real world implementations in this application area.
2. Universal representation of objects of virtual environment in the proposed algorithm allows one to handle interactions among objects by universal way. Such approach is not widely spread - systems of dynamic modeling as a rule introduce different groupings for object interactions, constraint object interactions and apply other additional requirements making modeling no uniform, complicating model description.
3. Using technique of message handling callbacks to compute reactions to collisions in the proposed algorithm is new in respect to this application area. It results in maximum versatility and low computational overhead.

The proposed algorithm satisfies two conditions formulated by us in the problem definition, namely:

- $O(N \log_2 N)$ and $O(N)$ computational complexity of the algorithm in its wide and narrow phases makes it applicable for real-time modeling
- using OOP technique to compute reactions to collisions makes it possible to extend set of reactions depending on modeling requirements for specific tasks.

Authors: *Sergey Vyatkin, Institute of Automation and Electrometry, Russia, e-mail: sivser@mail.ru, Tetiana Korobeinikova, Lviv Polytechnic National University Ukraine, e-mail: rom8591@gmail.com; Pavlo Mykhaylov, CEO 3D GNERATION GmbH, Germany, e-mail: pm@3dgeneration, Roman Chekhmestruk, 3D GENERATION UA, Vinnitsa, Ukraine, e-mail: Rc.ua@3dgeneration.com; Oksana Vodzinska, Kyiv National University of Technologies and Design, Ukraine, e-mail: oksiiv@ukr.net; Vasyl Ovcharuk, Vinnytsia National Technical University, e-mail: sofiyadem13@gmail.com; Andrzej Kotyra, Lublin University of Technology, Lublin, Poland, e-mail: a.kotyra@pollub.pl; Gulzhan Kashaganova, Institute of Information and Computational Technologies CS MES RK, Almaty, Kazakhstan and Kazakh-American University, Almaty, Kazakhstan, e-mail: guljan_k70@mail.ru; Zhazira Julayeva, Academy of Logistics and Transport, Almaty, Kazakhstan, e-mail: zhazj@mail.ru*

REFERENCES

[1] Kibble, T. W. B.; Berkshire, F. H., Classical Mechanics (fifth Ed.), Imperial College Press. (2004), pp. 236, ISBN 9781860944352.
[2] Goldstein H., Poole C.P. Jr., Safko, J.L.. Classical Mechanics (third Ed.). San Francisco, CA: Addison Wesley(2002).
[3] Dvorak, R.; Freistetter, Florian. Lagrange equations of the first kind, *Chaos and stability in planetary systems*, Birkhäuser. (2005), p. 24.
[4] Kozlovska, T. I., Kolisnik, P. F., Zlepko, S. M., et al., "Physical-mathematical model of optical radiation interaction with biological tissues," Proc. SPIE 10445, (2017).
[5] Haken, H., Information and self-organization (3rd Ed.). Springer. (2006), pp. 61. ISBN 3-540-33021-6.
[6] Shabana A. A., Computational continuum mechanics. Cambridge University Press, (2008), 118–119.
[7] Taylor J.R., Classical mechanics. University Science Books. (2005), 297.
[8] Padmanabhan T., Motion in a rotating frame, *Theoretical Astrophysics: Astrophysical processes* (3rd Ed.). Cambridge University Press (2000), 48, ISBN 0-521-56632-0.
[9] Chad G.R., Classical Mechanics of Nonconservative Systems, *Physical Review Letters*, 110 (2013), No. 17, 174301.
[10] Kvyetnyy R., et al., Improving the quality perception of digital images using modified method of the eye aberration correction, *Proc. SPIE* 10031 (2016), 1003113. DOI: 10.1117/12.2249164.
[11] Kvyetnyy R., Bunyak Y., Sofina O., et al., Blur recognition using second fundamental form of image surface, *Proc. SPIE* 9816 (2015), 98161A.
[12] Kvyetnyy R.N., Romanyuk O.N., Titarchuk E.O., et al., Usage of the hybrid encryption in a cloud instant messages exchange system, *Proc. SPIE* 10031 (2016), 100314S.
[13] Skublewska-Paszkowska, M., Smołka, J., Comparison of the selected motion interpolation methods, *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, 3 (2013), No. 3, 14-17. https://doi.org/10.35784/iapgos.1456
[14] Vyatkin S.I., Romanyuk O.V., Collision detection of solid objects, *Days of science in DonNTU: Proceedings of the VI International conference "Modeling and computer graphics"*, (Krasnoarmeysk, Ukraine, may 26-29, 2015), (2015), 113-124.