UDC 629-78

D. A. Khramov

COMPUTER SIMULATION OF HEAVY ROPE'S DEPLOYMENT WITH USE OF PHYSICS ENGINES

One trend of modern astronautics is the use of orbital extended systems. In 2007, from the YES2 satellite was deployed 30-km tether with re-entry capsule attached to the end. Such space tether system can be used to create interferometers with a large base, solar power systems, and for spacecraft deorbit [1, 2]. In connection with this problem the investigation of the dynamics of the deployment of such systems [3, 4] is an actual question.

Accounting of the effects of the space environment on the motion of an extended body in itself is quite a challenge. Therefore, at the initial stage of research it is advisable to simplify the originally formulation of the problem, considered the behavior of the system in a simple external conditions and explore deploying a heavy rope with fixed end in a field of constant gravity.

Tether system can be represented with point masses or rigid bodies connected to each other. Simulation of motion of such systems with the help of modern physical modeling packages (e.g., SimMechanics or MapleSim) is straightforward. Difficulties arise when solving problems in which the structure of the system changes during the movement. Adding bodies during deployment requires to stop the calculation and make adjustments to the scheme of the system.

Promising tool for solving such problems is physics engines — software libraries for computer simulation of motion of systems which consists of particles and bodies [5] . They allow to add or remove the body directly in the simulation runtime. Once created in connection with the development of computer games and 3D-modeling, these libraries are now widely used in virtual reality systems [6] and robotics. Overview of models, numerical methods, and algorithms used in physics engines contains in [7]. The most comprehensive list of such engines is shown on the site [8].

The aim of the paper is to review the main components of physics engines and creation on their basis the models of deployment of heavy rope.

Principles of simulations using physics engines

Unlike math libraries that implements some numerical methods, the physics engine as a software library enables the user to create objects such as “particle”, “body”, “constraint”, and manage their properties.

Modeling with use of the physics engine starts with the creation of the virtual “world” in which particles and bodies are placed. Body may be rigid or soft, with the specified characteristics of movement and weight. Furthermore, each body is assigned a specific shape (collision shape), which is then used in collision analysis.

Then user sets the forces and impulses that acting on the body, adds connection between bodies and calculates motion of system. The main objectives of the physics engine at this stage are the collision detection between bodies and calculations based on this information the bodies state at the next moment. The data of the bodies state is transferred to renderer.

At the end of the calculations the garbage collection is performed: removes constraints, bodies, and other objects.

Let us consider a time step. Primarily, performed collision detection between the bodies. It consists of broad and narrow phases. During broad phase, collision detection algorithm finds a pair of potentially interacting objects. It uses a simplified geometry of the system, for example, the body replaced the axis-aligned bounding box. Next comes the turn of the narrow phase, during which the engine must determine the actual point of contact of bodies, normal and penetration depth. Narrow phase is computationally much more expensive than broad, so the effectiveness of broad phase implementation are largely determine the performance of the engine with a large number of bodies.

Taking into account the collisions are detected engaged engine component called solver. The solver's task includes the calculation of the bodies velocity after the collision, and the correction of their positions in order to avoid penetration into each other.

At the final stage, the calculation of the new positions and velocities of the bodies are performed. Physics engines uses for it the traditional methods for the numerical solution of ordinary differential equations, in particular, the methods of Euler-Cromer, Verlet and Runge-Kutta.

More detail the internal structure and working principles of physics engines are described in [9–11].

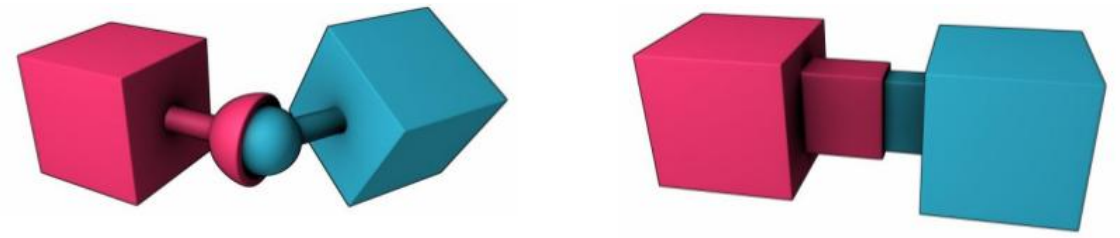
Model of rope

Let rope consists of pivotally coupled rigid bodies. In physics engines, the following three types of rigid bodies exists:

- dynamic rigid bodies are the conventional bodies, able to move and collide with others;
- static rigid bodies are motionless bodies, that just can collide with other bodies;
- kinematic rigid bodies are movable bodies with the user defined motion, and the collisions with other bodies can not change it.

In our case, need the dynamic bodies for rope and one static body to which the rope is attached.

To connect bodies we needs a revolute joint (fig. 1a), with one (in the planar case) or three (in spatial) rotational degrees of freedom, as well as a prismatic (sliding) joint, that has one translational degree of freedom (fig. 1b).



a) b)

Figure 1 — Connection between the bodies: a) revolute joint, b) prismatic joint [12]

In addition, the joints can be set with motors, i.e. forces governing mutual movement of connected bodies, and constraints (joint limits) for relative displacement/rotation of bodies. For deployment modeling we sets motor in a prismatic joint, that allow to push the body with a predetermined speed.

The scheme of deployed rope is shown in fig. 2.

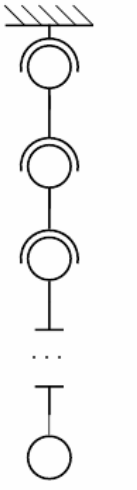


Figure 2 — The scheme of deployed rope

The algorithm of deployment

The rope's deployment algorithm consists of two main phases: initialization (fig. 3) and calculation step.

```
Create "world" and set gravity
Create static body "ground".
Create dynamic body "jointBody" and put it in the rope attachment point
Connect "ground" and "jointBody" with revolute joint
Create dynamic body "transBody"
Connect "jointBody" and "transBody" with prismatic joint, set joint motor
```

Figure 3 — The algorithm of deployment: initialization phase (pseudocode)

Upon completion of the initialization, the system takes the form shown in fig 4a. A rope consists of a single unit, comprising “jointBody” and “transBody”, connected by prismatic joint.

During deployment the bodies are moved apart at a predetermined speed. When the distance between them exceeds some fixed length (fig. 4b), the link is transformed: it removes prismatic joint (fig. 4c) and creates a new body “newBody” (fig. 4d). If the rope is not fully deployed, then “newBody” is connecting with "jointBody” by prismatic joint (fig. 4e), and deployment has been going on in the link “jointBody”–“newBody”. Now “newBody” plays the role of “transBody” (fig. 4f). If the rope has deployed, “newBody” and “jointBody” connected by revolute joint and the system takes the form shown in fig. 4h.

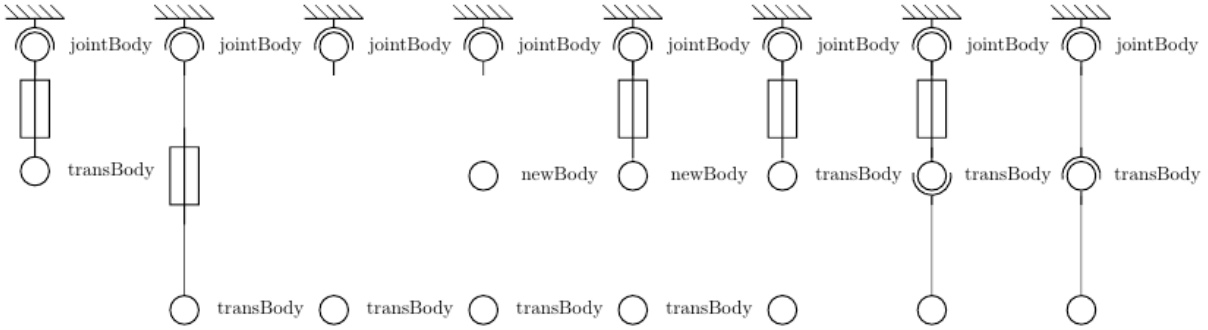


Figure 4 — The scheme of deployment

The calculation step of the deployment algorithm is shown in fig. 5.

```
Calculate the length of link
If the length of a link > "length"
    remove prismatic joint
    create a dynamic body "newBody"
    If the rope is not fully deployed
        connect "newBody" and "jointBody" by prismatic joint
    else
        connect "newBody" and "jointBody" by revolute joint
    connect "newBody" and "transBody" by revolute joint
    "transBody" = "newBody"
```

Figure 5 — The algorithm of deployment: calculation step (pseudocode)

The modeling of deployment

Consider the implementation of an algorithm of deployment by the examples of two physics engines — Box2D [13] and Bullet [14]. The source code of implementation can be found at [15].

Box2

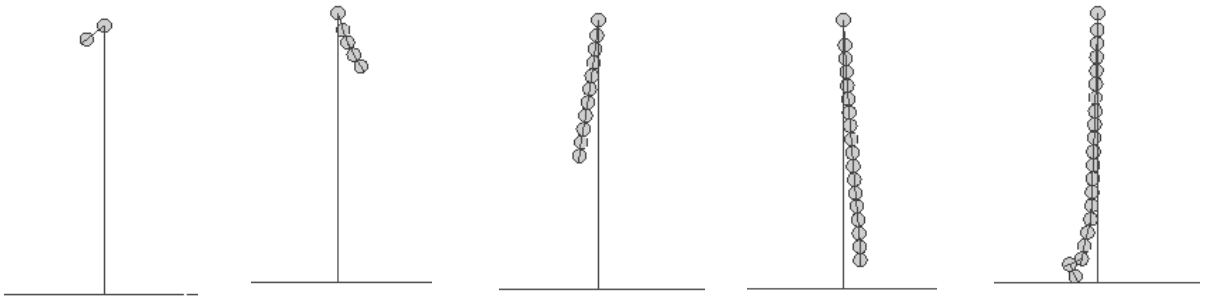
Box2D — free 2D-physics engine, developed by Erin Catto. Because to its simplicity for use and speed, Box2D is widely used in computer games (in particular, in Angry Birds), and educational applications.

Box2D is designed to simulate the dynamics of systems of rigid bodies. The bodies may be connected by various types of joints (a total of ten) provided with motors and delimited exposed to various forces, particularly gravitational forces, friction and impact. The engine is able to simulate the collision of bodies composed of convex polygons, circles and lines. Working with Box2D described in detail in [16].

Box2D is written in C++ and can run on any platform on which there is a C++ compiler. In addition, it has been ported to many programming

languages and software environments, including Java, C#, Adobe Flash, JavaScript and Delphi.

The simulation results of deployment, obtained with use of Box2D, are shown in fig. 6.



a) b) c) d) e)

Figure 6 — Animations frames of rope's deployment has modeled with use of Box2D

Bullet

Bullet — free 3D real-time physics engine, developed by Erwin Coumans. In addition to rigid bodies, the engine allows to simulate the movement of soft bodies, implements the methods of smoothed particle hydrodynamics, supports the CUDA technology and OpenCL.

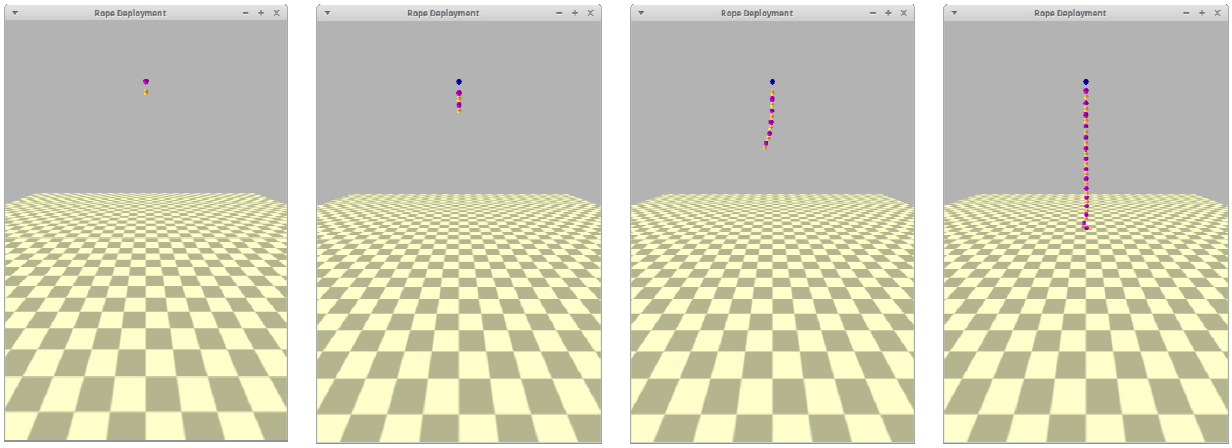
Bullet is widely used in video games (for example, in GTA 4, 5), films (Shrek 4), and 3D-modeling programs (Blender).

In the fig. 7 the results of modeling of rope's deploying with use of Bullet is shown.

Possibilities of verification of results

The above libraries are free and open source software, that allows the user to assess for themselves how to implement each component of the library, and, if necessary, correct errors or add new features.

Furthermore, it is possible to compare the results obtained using different engines. For this purpose there COLLADA Physics format and Physics Abstraction Layer (PAL) API.



a) b) c) d)

Figure 7 — Animations frames of rope's deployment has modeled with use of Bullet

COLLADA [17] — a standard format for data exchange between 3D-applications, based on XML. COLLADA standard consists of several parts, one of which — COLLADA Physics — responsible for the transmission of data about physical objects. COLLADA file is a snapshot of the virtual space that contains information about all the bodies are in it and the links between them. Standard is supported by the most famous physics engine, in particular, by Bullet. This enables to import COLLADA file, and perform an analogous simulation on another engine.

Another approach to data exchange is offered in PAL [18]. PAL is an interface to abstract physics engine through which the developer can use multiple engines or change between them, including directly in the calculations.

Conclusions

Using of physics engines allows the developer to work with objects that are closer to the problem domain — the multibody systems dynamics — compared to an approach, based on the use of general purpose programming languages, together with mathematical libraries.

Physical engines allows to develop without significant effort a program to rope's deployment and get results of calculations in real time. It should be noted that for most physical engines the speed of calculations is more important than accuracy. Therefore, in particular, many engines uses methods of numerical integration is not above the 2nd order. However, this lack of available engines is easy to fix by connecting additional math libraries.

Note that some physics engines allows to calculate the dynamics of deformable (soft) bodies. However, the methods on which these

calculations are based, are scantily described in manuals and require further study before their use in scientific purposes. Thus, the offered algorithm of rope's deployment uses only of pivotally coupled rigid bodies.

REFERENCES

1. Cartmell M. P. A review of space tether research / M. P. Cartmell, D. J. McKenzie // Progress in Aerospace Sciences. — 2008. — V. 44, N 1. — P. 1–21.
2. Волошенюк О. Л. Космические тросовые системы — перспективное направление космической техники и технологии / О. Л. Волошенюк, А. В. Пироженко, Д. А. Храмов // Космічна наука і технологія. — 2011. — Т. 17, № 2. — С. 32–44.
3. Dynamics of variable-length tethers with application to tethered satellite deployment / J. L. Tang, G. X. Ren, W. D. Zhu, H. Ren // Communications in Nonlinear Science and Numerical Simulation. — 2011. — V. 16, N 8. — P. 3411–3424.
4. Dynamical simulation of tether in orbit deployment / N. N. Smirnov, Yu. A. Demyanov, A. V. Zvyaguin, A. A. Malashin, A. A. Luzhin // Acta Astronautica. — 2010. — V. 67, N 3--4. — P. 324–332.
5. Jones M. T. Open source physics engines. Building believable worlds with open source [Электронный ресурс]. — Режим доступа : http://www.ibm.com/developerworks/opensource/library/os-physicsengines/os-physicsengines-pdf.pdf
6. An Evaluation of Open Source Physics Engines for Use in Virtual Reality Assembly Simulations / J. Hummel, R. Wolff, T. Stein, A. Gerndt, T. Kuhlen // Advances in Visual Computing. 8th International Symposium, ISVC 2012, Rethymnon, Crete, Greece, July 16–18, 2012, Revised Selected Papers, Part II. — 2012. — P. 346–357.
7. Interactive simulation of rigid body dynamics in computer graphics / J. Bender, K. Erleben, J. Trinkle, E. Coumans // EUROGRAPHICS 2012 State of the Art Reports. — Cagliari, Sardinia, Italy : Eurographics Association, 2012.
8. Physics Engines List [Электронный ресурс]. — Режим доступа : http://www.digitalrune.com/Support/Blog/tabid/719/EntryId/30/Physics-Engines-List.aspx

9. Millington I. Game Physics Engine Development / I. Millington. — New York : Morgan Kaufmann Publishers, 2010. — 553 p.
10. Bourg D. M. Physics for Game Developers / D. M. Bourg, B. Bywalec. — Cambridge : O'Reilly, 2013. — 551 p.
11. Eberly D. H. 3D Game Engine Architecture / D. H. Eberly. — New York : Morgan Kaufmann Publishers, 2005. — 736 p.
12. Bullet 2.82 Physics SDK Manual [Электронный ресурс]. — Режим доступа : http://bullet.googlecode.com/svn/trunk/Bullet_User_Manual.pdf
13. Box2D [Электронный ресурс]. — Режим доступа : http://www.box2d.org/
14. Bullet Physics Library [Электронный ресурс]. — Режим доступа : http://bulletphysics.org/
15. Компьютерное моделирование развертывания весомой нити с помощью физических движков [Электронный ресурс]. — Режим доступа : http://dkhramov.dp.ua/index.php?n=Sci.SciSoft
16. Box2D C++ tutorials [Электронный ресурс]. — Режим доступа : http://www.iforce2d.net/b2dtut/
17. COLLADA [Электронный ресурс]. — Режим доступа : http://collada.org/
18. PAL [Электронный ресурс]. — Режим доступа : http://www.adrianboeing.com/pal/index.html