COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2016; **27**:262–270
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1712

SPECIAL ISSUE PAPER

Predictable behavior during contact simulation: a comparison of selected physics engines

Se-Joon Chung[1]* and Nancy Pollard[2]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
[2] Robotics Institute, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

ABSTRACT

Contact behaviors in physics simulations are important for real-time interactive applications, especially in virtual reality applications where user's body parts are tracked and interact with the environment via contact. For these contact simulations, it is ideal to have small changes in initial condition yield predictable changes in the output. Predictable simulation is key for success in iterative learning processes as well, such as learning controllers for manipulations or locomotion tasks. Here, we present an extensive comparison of contact simulations using Bullet Physics, Dynamic Animation and Robotics Toolkit (DART), MuJoCo, and Open Dynamics Engine, with a focus on predictability of behavior. We first tune each engine to match an analytical solution as closely as possible and then compare the results for a more complex simulation. We found that in the commonly available physics engines, small changes in initial condition can sometimes induce different sequences of contact events to occur and ultimately lead to a vastly different result. Our results confirmed that parameter settings do matter a great deal and suggest that there may be a trade-off between accuracy and predictability. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
predictability; physics engine comparison; physics simulation; virtual worlds

***Correspondence**
Se-Joon Chung, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA.
E-mail: sejoonc@cs.cmu.edu

1. INTRODUCTION

Physics simulations are becoming more important for real-time interactive applications, especially for virtual reality applications where user's body parts are tracked and interact with the environment. Thankfully, we have many capable physics engines available for use such as Bullet Physics [1], Dynamic Animation and Robotics Toolkit (DART) [2], MuJoCo [3], and Open Dynamics Engine (ODE) [4].

However, it is often difficult to decide which one will be best for one's use case with so many choices of physics engines. A recent survey in the robotics community shows that even robotics researchers, who depend on physics simulations all the time, are divided among their choice of primary physics engine [5]. The survey also shows that robotics researchers are constantly seeking for better solutions, with many researchers trying out other engines to test their performance and some researchers abandoning previously used engines in favor of others.

This difficulty in choosing which physics engine to use is precisely what many physics engine comparison studies aim to shed some light on [6–9]. These studies compare various physics engines with regard to their speed, accuracy, and stability. Our work differs from those works primarily by focusing on predictability of the motion, in particular predictable variation in behavior over simulation tests having smoothly varying parameters.

In our comparisons, we tune individual engine's parameters to best match the analytical solution for a simple simulation. Then, we compare results for a more complex simulation where an analytical solution is hard to derive. In this way, we aim to compare all engines under the settings that produce the most accurate performance, because one engine's optimal settings may not be suitable for another.

In terms of predictability and smooth behavior, we found that while MuJoCo performed best in our first experiment, the same parameter setting caused it to not do well in our second experiment. Even though they did not produce simulations close to ground truth in the first experiment, ODE and DART produced the most predictable simulations in our second experiment.

2. RELATED WORKS

A number of researchers have performed physics engine comparisons, and this line of research is constantly evolving. Seugling and Rölin compared Newton Dynamics, ODE, and NovodeX (now PhysX) with the goal of finding a suitable physics engine to use in Virtools, a 3D authoring tool [6]. They compare and rank the three engines under nine-benchmark testing accuracy of friction force, accuracy of gyroscopic force, conservation of energy, constraint handling, and contact handling. Then, based on the sum of rankings in each criterion, they concluded that NovodeX had the best overall performance. In another comparison work by Boeing [7], more physics engines are compared with an additional focus on gaming technology. They compared seven physics engines (PhysX, Bullet, JigLib, Newton Dynamics, ODE, Tokamak, and True Axis) using integrator, material, constraint, collision, and stacking tests. They concluded that no one engine performed best at all tasks, but of the open-source engines, Bullet provided the best overall results for gaming. For simulation purposes, however, the most important property of the simulation should be determined first in order to select the best engine.

Most of the physics engines compared in the aforementioned studies are still undergoing active development and call for continued efforts in evaluating them. In a more recent study, Hummel *et al*. compare Bullet, Newton Dynamics, Havok, ODE, and PhysX with focus on high accuracy at interactive rates for assembly simulations [8]. They performed six benchmarks that test collision, constraint stability, interpenetration, and friction. They concluded there is no general physics engine that performs best for any given task. Furthermore, they suggest that physics engines optimized for games such as PhysX and Havok may provide less accurate simulations because of the optimizations they make for stability and performance. On the other hand, Bullet, Newton, and ODE were found to be good candidates for their assembly simulations. In another study by Peters and Hsu, four physics engines (Bullet, DART, ODE, and Simbody) included in Gazebo robot simulator are compared under four benchmarks [9]. They found certain engines may perform worse on specific benchmarks according to the contact model, joint damping, or coordinate system. However, for the most complex robot walking simulation, all of the engines produced similar trajectories.

Some have taken a different approach to physics engine comparison and focused more on benchmarks with complex models that can stress-test the engines. Erez *et al*. compared the self-consistency of Bullet, Havok, MuJoCo, ODE, and PhysX under identical simulation parameters in order to measure their integration error [10]. They found MuJoCo performed best on robotics-related tests, which it was designed for, while gaming engines won the gaming-related tests without a clear leader among them.

A different branch of works sought to compare the underlying contact algorithms directly without associations to specific physics engines. In this light, Drumwright and Shell have evaluated different methods of modeling contact in multibody simulation [11] and performance of linear complementary problem (LCP) solvers for rigid body contact problems [12]. Our work is complementary to their works in that we consider each engine as a whole when evaluating contact behaviors.

In order to derive an analytical solution to use as reference in our parameter tuning, we apply the rimless wheel model in McGeer's work [13] to a cube that we use throughout our comparisons. We then build up on their model to derive the relationship between angular velocity of the cube and its rolling behavior.

3. OVERVIEW OF PHYSICS ENGINES

We utilized a modified version of the physics simulation engine comparison framework used in the work of Erez *et al*. [10] for our comparisons. We made extensions to the framework to be able to apply active external forces. Unlike in the previous work, Havok and PhysX were not available to us, but we added DART, which is another popular physics engine. In the succeeding texts, we provide an overview of each engines included in our comparisons.

Bullet. Bullet Physics [1] is an open-source physics library developed by Erwin Coumans. The original Bullet is based on maximal coordinates and uses a sequential impulse (SI) solver for constraint resolution, but the recent version provides options to use various mixed linear complementarity problem (MLCP) solvers [14]. We explored using these MLCP solvers as well as the default SI solver. We used Bullet Physics version 2.83.7, which was released on 8 January 2016.

Bullet Multibody (Bullet MB). Bullet Physics includes a generalized coordinate approach based on Featherstone's algorithm [15] called multibody since version 2.82. We also include this multibody version of Bullet in our comparisons. Currently, the multibody version only supports SI solver that is an optimized implementation of projected Gauss–Seidel (PGS) method. The same 2.83.7 version of Bullet is used.

DART. DART [2] is an open-source physics library created by the Georgia Tech Graphics Lab and Humanoid Robotics Lab. Like Bullet MB, DART uses the generalized coordinate Featherstone approach. Although other MLCP solvers seem to exist in DART, the default solver used in this version of DART was Dantzig LCP solver. Because DART did not provide an interface to switch solvers at runtime, we used the default Dantzig LCP solver. We used DART version 5.1.1, which was released on 6 November 2015.

MuJoCo Euler. MuJoCo [3], which stands for Multi-Joint dynamics with Contact, was developed by Emo Todorov for Roboti LLC. MuJoCo is the only closed-source engine in our comparisons. MuJoCo is unique in the fact that it does not use an LCP solver to resolve constraints but formulates it as a convex optimization problem

[16,17]. In MuJoCo Euler, MuJoCo uses semi-implicit Euler integration method to update the velocity, but position is updated using the new velocity for stability. We used MuJoCo version 1.22, which was released on 26 November 2015.

MuJoCo RK. MuJoCo RK uses the same MuJoCo engine as MuJoCo Euler, but uses fourth-order Runge–Kutta method instead of Euler's method for integration.

ODE. ODE [4] is an open-source physics library originally developed by Russell Smith. Like the original Bullet Physics, it uses a Cartesian coordinate-based approach. ODE provides both a Dantzig LCP solver and successive over-relaxation PGS LCP solver [18]. We chose to use the successive over-relaxation PGS solver because Dantzig solver is slower and sometimes fails to find a solution. We used ODE version 0.14, which was released on 18 December 2015.

4. ROLLING CUBE

In order to compare the performance of various physics engines, we performed a cube rolling experiment that can be solved analytically by simplifying it as a 2D problem. This simplification holds for a rotation about one of its edges.

4.1. Condition for Cube Rolling Over

Given a cube that is rolling down a slope on one of its edges such that the edge becomes its rotation axis with initial angular velocity ω_0 around that axis, we can compute the condition of initial angular velocity ω_0 such that it would roll over and potentially trigger another rotation about a different edge. In order for this to happen, the angle θ between the cube and the surface must be able to surpass $\theta_0 - \theta_s$ before gravity can pull it down on another side (Figure 1). Here, θ_s denotes the slope angle, and θ_0 denotes half of the angle between point A, center of mass (point m), and point B. In our analysis, we assume that energy is conserved while cube is rotating about point B.

When the cube is initialized with rotational velocity ω_0 about point B (Figure 2(a)), its initial total energy is the

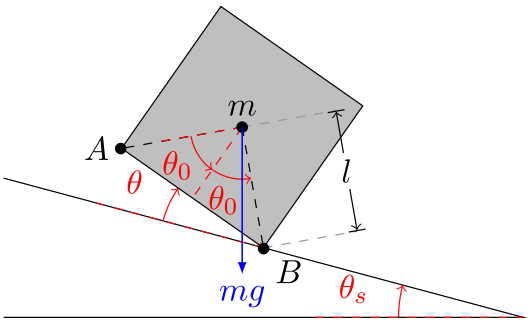

Figure 1. Gravity force acts on the cube while it rotates about point *B*.

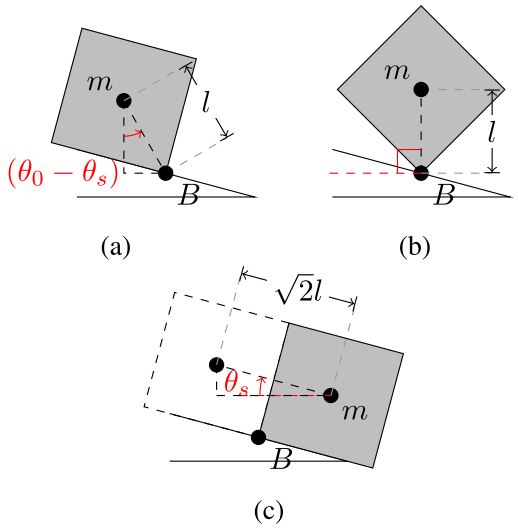

Figure 2. Cube rolling cycle around point *B*. (a) Initial configuration, (b) peak configuration, and (c) final configuration.

sum of rotational kinetic energy $\mathrm{KE}_{\mathrm{rotational}} = \frac{1}{2} I_B \omega_0^2$ and potential energy with respect to point B $\mathrm{PE}_B = mgl\cos(\theta_0 - \theta_s)$, where $g = 9.81\,\mathrm{m/s^2}$ is the standard acceleration due to gravity. Here, the moment of inertia I_B about point B can be is calculated using square's radius of gyration $r_{gyr} = \frac{1}{\sqrt{6}l}$ and parallel axis theorem. We normalized r_{gyr} by l for simplification.

$$I_B = \left(r_{gyr}^2 + 1\right) ml^2 \tag{1}$$

When the cube reaches the peak (Figure 2(b)), its total energy must be greater than the potential energy at the peak in order for it to tip to the other side and continue rolling. Equation 2 shows the derivation of the condition on initial angular velocity ω_0 for the cube to keep rolling.

$$\begin{aligned} \textstyle\sum E_{\mathrm{initial}} &> \mathrm{PE}_{\mathrm{peak}} \\ \tfrac{1}{2} I_B \omega_0^2 + mgl\cos(\theta_0 - \theta_s) &> mgl \\ \omega_0 &> \sqrt{\frac{2g(1-\cos(\theta_0 - \theta_s))}{l\left(r_{gyr}^2 + 1\right)}} \end{aligned} \tag{2}$$

Furthermore, if the cube does roll over, the new angular velocity right before the next impact can be calculated using a similar energy-based derivation. Because the change in height from the initial configuration will be $\sqrt{2}l\sin(\theta_s)$ as shown in Figure 2(c), the new angular velocity ω can be calculated as in equation 3.

$$\begin{aligned} \tfrac{1}{2} I_B \omega_0^2 + \sqrt{2}l\sin(\theta_s)mg &= \tfrac{1}{2} I_B \omega^2 \\ \omega &= \sqrt{\frac{I_B\omega_0^2 + 2\sqrt{2}l\sin(\theta_s)mg}{I_B}} \end{aligned} \tag{3}$$

DOI: 10.1002/cav

Table I. Parameters, their functions, default values, and candidate values.

Parameter	Function	Default value	Candidate values
μ	Determines the maximum amount of friction that can be exerted at contact points.	N/A	1.0, 2.0, 3.0
ERP	Specifies what proportion of the joint error will be fixed during the next simulation step.	0.2	0.0, 0.1, 0.2, 0.5, 1.0
CFM	Determines the amount constraint is allowed to be violated by an amount proportional to CFM times the restoring force that is needed to enforce the constraint.	0.0	0.0, 0.1, 0.2, 0.5, 1.0
Number of iterations	Number of iterations performed in the engines' LCP solver. Higher iteration means more accuracy but takes more time.	10 for Bullet, 20 for ODE	10, 20, 30, 50, 100
MLCP	Determines which MLCP solver in Bullet is used when resolving the constraints. This parameter does not apply to Bullet MB, which only uses a different multibody constraint solver.	Sequential impulse	Sequential impulse, Dantzig, Lemke, projected Gauss–Seidel
Contact surface layer	Contacts in ODE are allowed to sink into the surface layer up to the given depth before coming to rest. The default value is zero. Increasing this to some small value (e.g., 0.001) can help prevent jittering problems because of contacts being repeatedly made and broken.	0.0	0.0, 0.001, 0.01, 0.1
dmin	Together with dmax and width, parameterizes sigmoid system impedance function d(r) in MuJoCo.	0.9	0.85, 0.9, 0.95, 1.0
dmax	Together with dmin and width, parameterizes sigmoid system impedance function d(r) in MuJoCo.	0.95	0.9, 0.95, 1.0
Width	Together with dmin and dmax, parameterizes sigmoid system impedance function d(r) in MuJoCo.	0.001	0.0005, 0.001, 0.002
timeconst	Inversely scales the stiffness constant in MuJoCo and also inversely scales the damping ratio by square of timeconst. It should be set to at least two times larger than the simulation time step, which is 1 ms in our case.	0.02	0.01, 0.02, 0.03
dampratio	Inversely scales the damping ratio in MuJoCo by square of dampratio. Less than 1 is under-damped, larger than 1 is over-damped, and 1 is critically damped.	1.0	0.5, 1.0, 1.5

μ, coefficient of friction; ERP, error reduction parameter; CFM, constraint force mixing; MLCP, mixed linear complementarity problem; LCP, linear complementary problem; Bullet MB, Bullet Multibody; ODE, Open Dynamics Engine.

4.2. Conservation of Angular Momentum

When the cube collides with the ground and it begins to roll on a different side, the cube loses some of its energy. We can compute the new angular velocity after the collision using conservation of angular momentum [13]. We assume the impact with the ground to be inelastic and without sliding.

The angular momentum before collision is given by equation 4.

$$\begin{aligned} L^- &= ml^2\omega^- \cos 2\theta_0 + I_{cm}\omega^- \\ &= \cos 2\theta_0 ml^2\omega^- + r_{gyr}^2 ml^2\omega^- \\ &= \left(\cos 2\theta_0 + r_{gyr}^2\right) ml^2\omega^- \end{aligned} \tag{4}$$

The angular momentum after collision is given by equation 5.

$$\begin{aligned} L^+ &= \left(I_{cm} + ml^2\right)\omega^+ \\ &= \left(r_{gyr}^2 ml^2 + ml^2\right)\omega^+ \\ &= \left(1 + r_{gyr}^2\right) ml^2\omega^+ \end{aligned} \tag{5}$$

Because angular momentum must be conserved before and after the collision, equating equations 4 and 5 yields the angular velocity after the collision ω^+ as a function of angular velocity before the collision ω^- (equation 6). In our case, the cosine term drops because $2\theta_0 = 90°$.

$$\omega^{+} = \left(\frac{\cos 2\theta_0 + r_{gyr}^2}{1 + r_{gyr}^2} \right) \omega^{-} \tag{6}$$

5. RESULTS AND DISCUSSION

In our experiments, we used a cube with each edge measuring 1 m, mass of 1 kg, and uniform density. Please also look at http://www.cs.cmu.edu/~sejoonc/papers/PredictablePhysics.html for video results and additional experiments.

5.1. Rolling Downhill

The goal of the first experiment was to find a set of parameters for each engine to best match the analytical solution from section 4. In this experiment, the cube was initialized with an angular velocity of ω_0 around the bottom right edge and rolled down a slope. We used an initial angular velocity of $\omega_0 = 2.4688$ rad/s, which is the largest initial angular velocity that should not tip over the cube on a flat ground. Each set of experiments consisted of 31 different slope angles between 0° and 30° in 1° increments.

This set of experiments with various slope angles was then repeated for each unique set of parameters in an exhaustive manner. Because sampling over the entire space of possible parameters would be overly time-consuming, some reasonable candidate values for each parameters were chosen around the recommended range of values in each engine's documentation. However, some parameters were not included in the parameter tuning to provide grounds for fair comparison. Time step was set to 1 ms for all engines, and coefficient of restitution was set to 0 to meet our assumption of inelastic collisions. Coefficient of friction was not fixed because we only assumed non-slipping behavior without any assumptions on the magnitude of friction force. Table I explains what each parameter does and the candidate values that were chosen for our search.

Initially, the best set of parameters for each engine was chosen as the set that best matched the number of cube rolls in ground truth predictions. However, we found that only matching the number of rolls yielded parameters for some engines where the cube wobbled wildly as it rolled down. Therefore, the criterion for the best set of parameters was changed to also account for minimum average deviation angle of the rotation axis from the y-axis ("wobble"). Scores were assigned by normalizing the number of cube roll matches and average deviation angle in the range of 0 to 1 with 1 being the best; then the set of parameters with the maximum sum of scores was chosen.

Table II lists relevant parameters for each engine and the final values chosen. It is worth noting that DART did not have obvious API or documentation for tweaking internal parameters, so only coefficient of friction was searched. Bullet and Bullet MB's parameters came close to default values with the number of iterations being a little bit higher.

Table II. Engines, their parameters, and final values chosen.

Engine	Parameters	Final Value
Bullet	μ	1.0
	ERP	0.2
	CFM	0.0 (no effect)
	Number of Iterations	20
	MLCP solver	Projected Gauss–Seidel
Bullet MB	μ	1.0
	ERP	0.2
	CFM	0.0 (no effect)
	Number of Iterations	30
DART	μ	1.0
MuJoCo Euler	μ	3.0
	dmin	0.85
	dmax	0.95
	Width	0.002
	timeconst	0.02
	dampratio	0.5
MuJoCo RK	μ	3.0
	dmin	0.85
	dmax	0.9
	Width	0.002
	timeconst	0.02
	dampratio	0.5
ODE	μ	2.0
	ERP	0.0
	CFM	1.0
	Number of Iterations	20
	Contact Surface Layer	0.0 (no effect)

μ, coefficient of friction; ERP, error reduction parameter; CFM, constraint force mixing; MLCP, mixed linear complementarity problem; Bullet MB, Bullet Multibody; DART, Dynamic Animation and Robotics Toolkit; ODE, Open Dynamics Engine.

MuJoCo Euler and MuJoCo RK preferred to have a wider width with lower dmin for the impedance function, which means thicker and softer contact layer around the objects. Both also preferred to be under-damped, which suggests that MuJoCo had to relax its constraints and preserve more energy in order to match the ground truth. ODE also seemed to prefer soft constraints with constraint force mixing value of 1.0. It was unexpected that ODE would prefer no error reduction with error reduction parameter of 0.0, but this yielded the best result.

Figure 3 shows the overall results with the chosen parameters. In Figure 3(a), each dot's color represents the number of times the cube rolled. Empty spaces repre-

DOI: 10.1002/cav

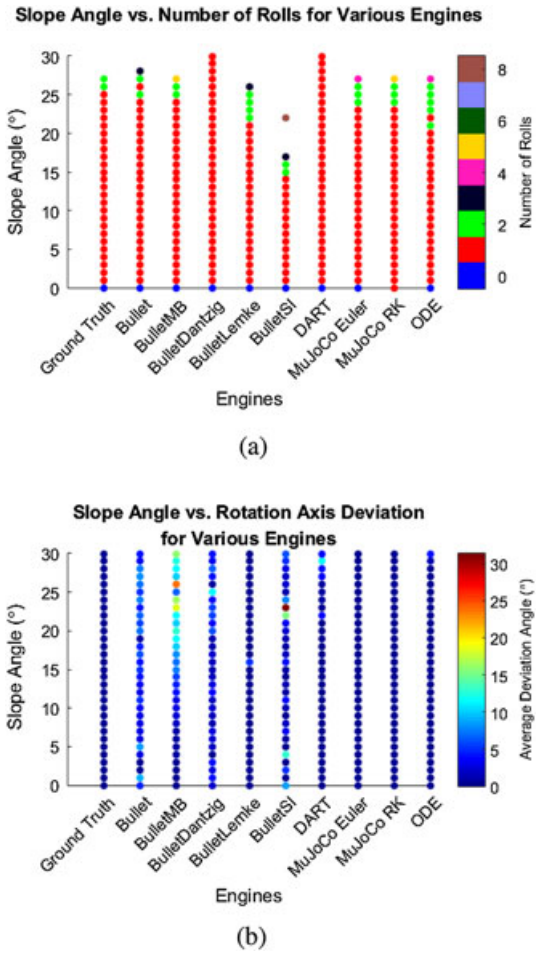

Figure 3. Cube downhill rolling simulation results with final parameters. (a) Number of times the cube rolls down the slope for various slope angles and engines and (b) average deviation angle of rotation axis from the *y*-axis.

sent simulations in which the cube rolled indefinitely. In Figure 3(b), each dot's color represents the average deviation angle of rotation axis from the *y*-axis. Even though the cube is initialized to only rotate about the *y*-axis, contact with the ground induces the rotation axis to tilt in many simulations. Therefore, we see simulations with less rotation axis deviation as more stable simulations. To investigate the effect of LCP solver used, we also included Bullet with the same parameter settings, but with each of the LCP solvers.

Overall, all of the physics engines had very distinct results. We found Bullet MB to have the best match in the number of rolls, but it suffered from instability in the rotation axis. In terms of stability, MuJoCo RK had the least rotation axis deviation. The best overall result was achieved by MuJoCo Euler with a good balance between the two metrics. Even though we expected a monotonically increasing number of rolls as the slope angle increases, some engines such as Bullet and ODE had instances where this was not true. In Bullet's case, the cube suddenly lost most of its angular velocity after the first collision on the 26° slope. In ODE's case, the problem was found to be on the 21° slope, which should have rolled only once, but rolls twice because of not enough reduction in angular velocity from the first impact. These can be seen as negative impacts to the engines' predictability.

Among Bullet's different LCP solvers, it was surprising that Bullet's SI solver and PGS solver yielded different results, because SI solver was documented to be an optimized version of the PGS solver. We found Bullet's PGS solver yielded not only better matches but also significantly less wobble than the SI solver under the same parameters. Bullet's Lemke solver resulted in the most stable simulations although it was not able to match the number of rolls very well. It is also worth noting that Bullet's Dantzig solver results closely matched that of DART, which also uses a Dantzig solver, even though there is a fundamental difference in the coordinate systems the two engines use (maximal versus generalized).

We found MuJoCo to be extremely stable within the combinations of our tested parameter ranges. For MuJoCo RK, the maximum average deviation angle of rotation axis among all sets of parameters was found to be 0.3507°. MuJoCo Euler's maximum average deviation angle was found to be a little bit higher at 1.2315°, but still much lower than other engines' maximum average deviation angles, which were 64.4543° for Bullet, 44.9917° for Bullet MB, 4.6051° for DART, and 25.4984° for ODE. Although MuJoCo gave excellent results and stability, it is interesting to notice that despite having not enough initial velocity to roll over at 0° slope, the fourth-order Runge–Kutta integrator was the only one that does. This did not happen for other engines which all use Euler's method for integration.

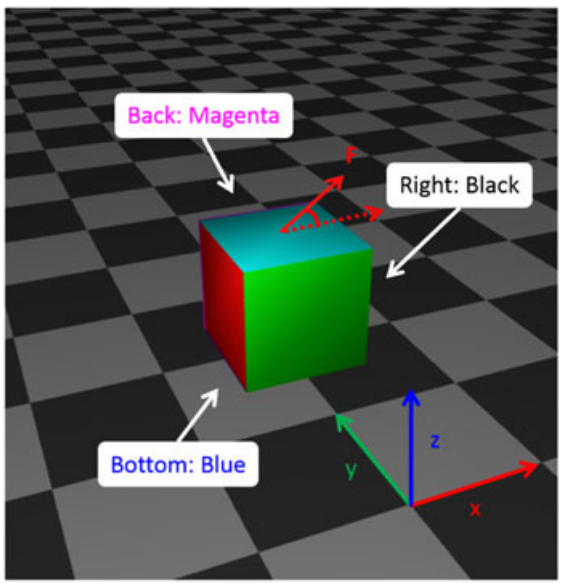

Figure 4. Cube in 3-D and the applied force.

5.2. Rolling on a Flat Ground in Various Directions

As a more extensive test for predictability in simulations with contacts, we performed a cube rolling test on a flat plane in all directions, not just in directions orthogonal to one of the cube edges. We did this by applying varied magnitude of force at the center of cube's top face for 0.1 s in the specified direction. The direction is measured in rotation about the z-axis, starting from the x-axis direction. For convenience, the cube was color coded on each side with the teal side on top initially (Figure 4).

In this test, for a fixed direction of force, we expect to see the cube landing on a certain side until enough force has been applied for it to be able to roll once more to an adjacent side. Conversely, for a fixed magnitude of force, we can expect to see the cube rolling on a certain edge until the force direction becomes closer to an adjacent edge, in which case it will roll on that adjacent edge. As a result of these two expectations, we expect to see clearly defined C-shaped regions of each side of the cube, interleaved in a

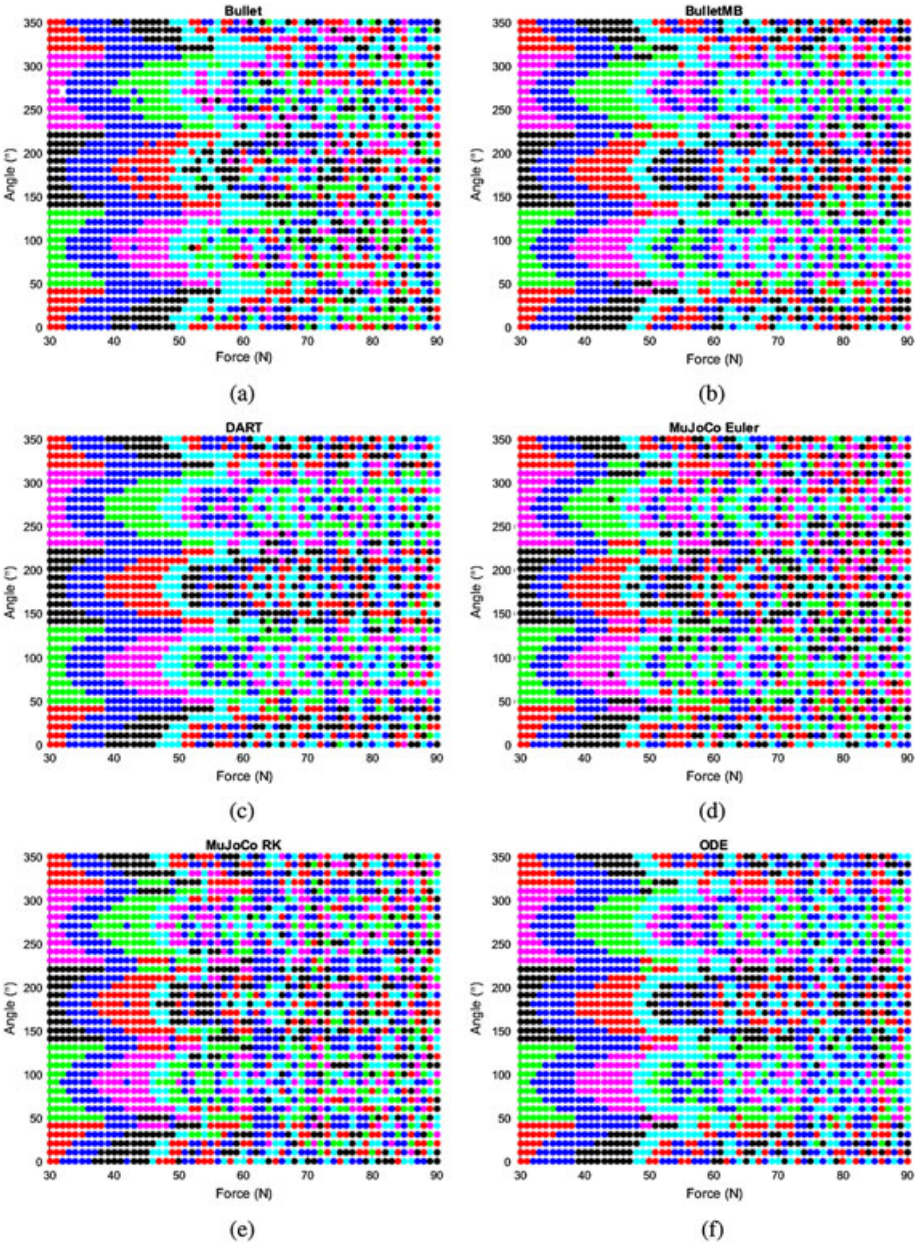

Figure 5. Cube side that ends up on top after various magnitude of force has been applied in various directions. (a) Bullet, (b) Bullet Multibody, (c) Dynamic Animation and Robotics Toolkit, (d)MuJoCo Euler, (e) MuJoCo RK, and (f) Open Dynamics Engine.

DOI: 10.1002/cav

way such that regions corresponding to adjacent sides are neighboring each other.

Figure 5 shows the actual results of this test for all of the engines. We can see that our expectations are confirmed in the lower ranges of the force with magnitude less than 50 N. For higher magnitude of forces, different smaller patches are dominant. We think this is a direct result of the cube spending some time airborne.

Among the different results, we can clearly see that some engines had more irregularity in their outcome than others, especially in the upper ranges. Bullet had the most unpredictable behavior. In one instance, the cube got stuck while standing on one of its edges at 32 N and 270° (hence the empty dot). We can also see the asymmetry and irregularities in the magenta, red, and green regions between 40 and 56 N range. Bullet MB fared much better than its maximal coordinates counterpart but suffered from occasionally falling on an orthogonal edge when force is applied in diagonal directions (e.g., isolated points along the 45 N). DART was one of the engines with more predictable behaviors. Points along directions orthogonal to each of its edges and 55 N may seem like anomalies, because these are points where cube rolls more times than its neighboring points. This type of phenomenon seems to happen in situations where the cube becomes airborne: The exact landing configuration will strongly affect collision response forces and angular momentum. MuJoCo Euler and RK, which were found to be very stable in the first experiment, suffered instability in this experiment because of the relaxed damping setting chosen during the parameter search. Because the system was under-damped, the cube suffered instabilities from too much restitution. ODE came out to have the most predictable result in this experiment, which was surprising given its unusual parameter setting.

While tuning the various parameters, we have observed many instances of trade-off between accuracy in the first experiment and the predictability of results in this experiment. One example is shown in Figure 6. By simply changing the damping parameter to 1.0 for MuJoCo RK, matches in number of rolls are reduced, but average axis deviation angle decreases, and predictability of results in the second experiment appears much better.

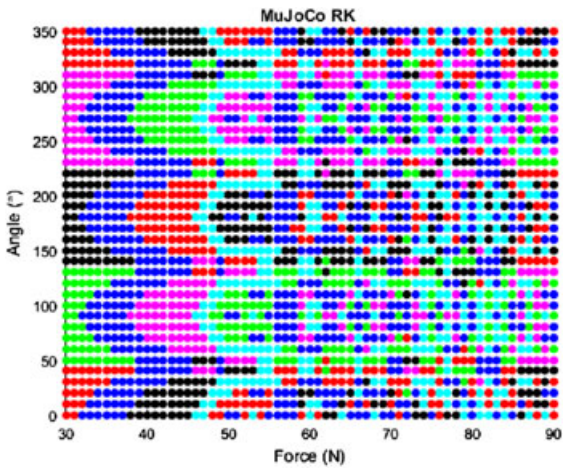

Figure 6. MuJoCo RK with damping set to 1.0.

Other possible reasons for the difference among the engines include differences in the collision detection algorithm and the friction model used. However, the details of actual implementation are not exposed to the user as a tunable parameter for most engines and were not included in this study.

6. CONCLUSION AND FUTURE WORK

We have provided an extensive comparison of contact simulations using Bullet Physics, DART, MuJoCo, and ODE. First, we tested a cube downhill rolling simulation where an analytical solution was derived through the application of classical mechanics and the rimless model from McGeer [13]. Then, knowing the importance of parameter tuning for the engines' performance, we tuned individual engine's parameters separately to match the analytical solution as closely as possible without sacrificing the stability of the engines. From the comparison of results with tuned parameters and ground truth, we found that while Bullet MB had the greatest number of matches in the number of rolls and MuJoCo RK had the least rotation axis deviation, MuJoCo Euler achieved a good balance between the two. In the case of Bullet and ODE, we found cases of unpredictable behavior where even though the slope angle was increased, the cube rolled fewer times than with a smaller slope angle.

In our second experiment, we performed a more extensive test for predictability by rolling the cube on a flat plane in all directions with varied magnitude of force. Here, the expected result for an engine with good predictability was to show a clearly repeating pattern in which side ends up on top after the cube came to a stop. Somewhat differently from the findings in the previous experiment, we found that DART and ODE showed nice regular patterns that are favorable to predictability.

As extensive as our comparisons were, there is room for expansion in this area. First, our study can be extended to other popular physics engines such as Havok, PhysX, or Simbody. It would be also interesting to compare SCISim [19], which simultaneously guarantees symmetry preservation and kinetic energy conservation, while allowing breakaway. Next, the set of complex simulations without an analytical solution can be extended to include animated characters or robots that have internal forces arising from actuation in addition to external contact forces. Last but not the least, our main motivation was to find which physics engines provide good predictability for sensitive simulations such as virtual reality applications and iterative learning processes. It would be encouraging to see important criteria from these fields be reflected back in future physics engine comparison works. In particular, we would like to see predictability added to the standard suite of benchmarks used for physics engine comparisons.

ACKNOWLEDGEMENTS

We would like to thank Emo Todorov, Tom Erez, Vikash Kumar, and Yuval Tassa for providing us with the source code to their physics engine comparison framework, a trial license for the MuJoCo engine, and invaluable advice.

REFERENCES

1. Real-time physics simulation. Available from: http://bulletphysics.org [Accessed on 2 February 2016].
2. DART. Available from: http://dartsim.github.io/ [Accessed on 2 February 2016].
3. MuJoCo. Available from: http://www.mujoco.org/ [Accessed on 2 February 2016].
4. Open Dynamics Engine - home. Available from: http://www.ode.org/ [Accessed on 2 February 2016].
5. Ivaldi S, Peters J, Padois V, Nori F. Tools for simulating humanoid robot dynamics: a survey based on user feedback. In *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Madrid, Spain, November 2014; 842–849.
6. Seugling A, Rölin M. Evaluation of physics engines and implementation of a physics module in a 3d-authoring tool. *Master's Thesis*, Umeå University, SE-901 87 UMEÅ SWEDEN, 2006.
7. Boeing A, Bräunl T. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. ACM, New York, NY, USA, 2007, GRAPHITE '07; 281–288.
8. Hummel J, Wolff R, Stein T, Gerndt A, Kuhlen T. An evaluation of open source physics engines for use in virtual reality assembly simulations. In *Advances in Visual Computing*, Vol. 7432. Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; 346–357.
9. Comparison of rigid body dynamic simulators for robotic simulation in Gazebo. Available from: http://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/04/roscon2014_scpeters.pdf [Accessed on 5 February 2016].
10. Erez T, Tassa Y, Todorov E. Simulation tools for model-based robotics: comparison of bullet, Havok, MuJoCo, ODE and PhysX. In *IEEE International-Conference on Robotics and Automation, ICRA 2015*, Seattle, WA, USA, 26-30 May, 2015; 4397–4404.
11. Drumwright E, Shell DA. An evaluation of methods for modeling contact in multibody simulation. In *2011 IEEE International Conference on Robotics and automation (ICRA)*, Shanghai, China, May 2011; 1695–1701.
12. Drumwright E, Shell DA. Extensive analysis of linear complementarity problem (LCP) solver performance on randomly generated rigid body contact problems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Algarve, Portugal, October 2012; 5034–5039.
13. McGeer T. Passive dynamic walking. *Int. J. Rob. Res.* 1990; **9**(2): 62–82.
14. Exploring MLCP solvers and featherstone. Available from: http://goo.gl/84N71q [Accessed on 4 February 2016].
15. Featherstone R. *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc.: Secaucus, NJ, USA, 2007.
16. Todorov E. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014; 6054–6061.
17. MuJoCo overview. Available from: http://www.mujoco.org/book/overview.html [Accessed on 4 February 2016].
18. physics_ode/ODE – ROS Wiki. Available from: http://wiki.ros.org/physics_ode/ODE [Accessed on 4 February 2016].
19. SCISim: a 2D and 3D rigid body simulation framework with a focus on preserving core physical properties. Available from: https://github.com/breannansmith/scisim [Accessed on 16 March 2016].

AUTHORS' BIOGRAPHIES

Se-Joon Chung is a PhD student in the Computer Science Department at Carnegie Mellon University (CMU), Pittsburgh, PA, USA, where he is advised by Professor Nancy S. Pollard. He has received a bachelor's degree in computer engineering at the University of Illinois at Urbana-Champaign where he also received the Bronze Tablet Award, E. C. Jordan Award, and A. R. "Buck" Knight Award. His research interests include hand pose prediction and virtual object manipulation for virtual reality applications.

Nancy S. Pollard has received the PhD degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1994. She is currently an associate professor at the School of Computer Science, Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Before joining CMU, she was an assistant professor at Brown University. Dr. Pollard received the National Science Foundation CAREER Award in 2001 and the Okawa Research Award in 2006.

DOI: 10.1002/cav