Inpher: Inferring Physical Properties of Virtual Objects from Mid-Air Interaction

Søren Qvist Jensen
Aarhus University
Aarhus, Denmark
soeren.qvist.jensen@post.au.dk

Andreas Fender
Aarhus University
Aarhus, Denmark
andreasfender@cs.au.dk

Jörg Müller
University of Bayreuth
Bayreuth, Germany
Joerg.Mueller@uni-bayreuth.de

ABSTRACT

We present Inpher, a virtual reality system for setting physical properties of virtual objects using mid-air interaction. Users simply grasp virtual objects and mimic their desired physical movement. The physical properties required to fulfill that movement will then be inferred directly from that motion. We provide a 3D user interface that does not require users to have an abstract model of physical properties. Our approach leverages users' real world experiences with physics. We conducted a bodystorming to investigate users' mental model of physics. Based on our iterative design process, we implemented techniques for inferring mass, bounciness and friction. We conducted a case study with 15 participants with varying levels of physics education. The results indicate that users are capable of demonstrating the required interactions and achieve satisfying results.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords

Mid-air interaction; performance-based; infer physical properties

INTRODUCTION

Real-time physics engines create plausible motion of virtual objects without the need for predefined animations. This allows for flexible interactive virtual environments. Developers of these interactive environments typically input physical properties based on well established physical models. However, this requires a certain level of knowledge about these models to input meaningful numbers and to have an intuition about the effects of adjusting these. Therefore, setting physical properties, as well as other aspects of designing a virtual environment, is typically a demanding process during development. Many content creation tools have been developed for users with no actual content creation background. These simple, but powerful tools aim at the easy creation of virtual 3D worlds [20], models [19], and animations [9] to enable creativity of users without requiring domain-knowledge. However, in all of these tools, editing physical behavior of virtual objects is typically not addressed and is instead based on predefined sets of physical properties. Users have very little control over physical behavior, which limits the expressiveness of these tools.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

 ISBN 978-1-4503-5620-6/18/04…$15.00

DOI: https://doi.org/10.1145/3173574.3174104

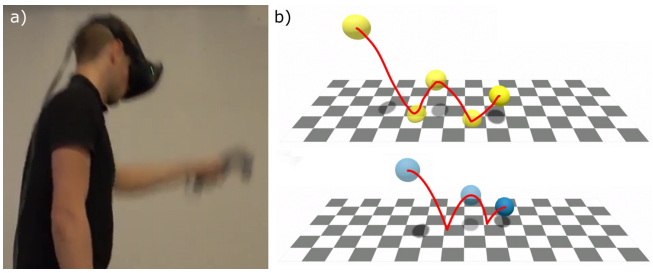

Figure 1. Inpher makes it possible to define physical properties like bounciness of virtual objects through mimicking their physical behavior. (a) The user is equipped with virtual reality goggles and controllers for 3D input. (b) Users can grasp virtual objects and describe a physical motion to define the object's physical behavior (top). The free-flight physical motion of the object resembles the user's input curve (bottom).

We present Inpher, a virtual reality system for setting physical properties using mid-air interaction. Equipped with VR goggles (e.g., HTC Vive [10]), users are situated in a virtual environment. Virtual objects react physically to interactions with each other and the user. To define their physical behavior, virtual objects can be grasped and *trained*, i.e., the user mimics a physical motion to indicate, how it is supposed to behave when colliding, bouncing etc. After training, the system infers the physical properties of the object like bounciness, mass and friction so that its physical behavior resembles the user's intention as closely as possible. Figure 1 shows an example for inferring bounciness. We contribute (1) An approach for setting physical properties of dynamic virtual objects using mid-air interaction, (2) a proof-of-concept system that infers bounciness, mass and friction of virtual objects, and (3) a case study that shows that users with little to no knowledge about physics were able to set desired physical properties.

RELATED WORK

Much of our work is based on classical mechanics, which can be found in physics literature (e.g. [2]). The idea of

inferring physical properties from motion of real objects is well explored [6, 3, 13, 21]. These approaches use video and motion capture data of real objects as input to accurately determine their mechanical properties.

We focus our attention on the challenges emerging from performance based 3D user input. Instead of capturing objects, users perform the physical motion to express their intention and the properties of objects that are not physically available or even realistic. This allows for flexibility and expression, but it also implies that the input is limited by human capabilities and is generally not physically accurate. Further challenges are the design of appropriate interaction techniques to express these motions. To the best of our knowledge, there is no system with the same overall objectives as ours. However, closely related systems focus on the inference of physical properties for animation. The system of Popović et al. [18] lets animators define various positions and orientations of a virtual object for specific points in time. The system then infers the physical properties necessary to meet the constraints in a physically plausible motion. Another system of Popović et al. [17] enables users to sketch a motion through mid-air interaction or by moving a physical prop. The system transforms the sketched trajectories into physically plausible movements. Laszlo et al. [12] present a system that creates physically plausible 2D animations based on mouse movements. All of these animation systems infer properties for particular motions or sub-movements, whereas we aim at inferring persistent physical properties. Further related approaches can be found in the field of human-robot interaction in particular in *learning by demonstration* techniques. These techniques and our approach share the common goal to let non-domain-experts demonstrate desired behavior. (see Argall et al. [4] for an overview). Examples of these techniques include demonstration by moving the robot's joints directly [7] or by using user-attached sensors or markers [11, 16].

We contribute to the literature by proposing an approach to enable novice users to interactively define physical properties of virtual objects.

DESIGN PROCESS

The core idea is inspired by the bodily expression of mechanical properties based on intuition and experience, instead of abstraction and knowledge about mechanics. For instance, a kid playing with action figures mimics the movement of a heavy imaginary character by making the figurine move slowly with impactful steps. In contrast, the kid moves a figurine of a lightweight character with a fast acceleration. The kid has no knowledge about physical abstractions, but implicitly communicates the intended mass of the characters, independently of their actual mass of the figurines.

We started by investigating the mental model of users regarding physical behavior of objects. To do so, we invited 9 participants (1 female) with ages ranging from 24 to 37 to a bodystorm [15]. We provided several physical props, like balls, ropes etc. to participants. Participants were asked to express different physical properties by grasping and moving the props, letting them collide etc. There was a wide range

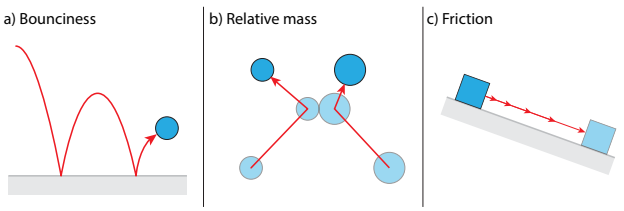

Figure 2. Supported physical properties of our proof-of-concept system. Users train virtual objects (blue) by grasping and moving them. The respective property value is calculated based on certain characteristics of the (red) trajectories describing the demonstrated motion of users. Bounciness (a) is a repeating down-and-up motion. Relative mass (b) is the the collision of two objects. Friction (c) is a sliding-down movement on an inclined surface.

of approaches and potential interaction techniques, which influenced our prototype system and inspired future work as discussed later.

SYSTEM

We implemented a proof-of-concept system, which supports a selection of physical properties, namely bounciness, mass and dynamic friction. Figure 2 provides an overview. The core idea is not limited to these three properties. They are an initial exploration of the design space that our approach defines. Users can freely move within a virtual environment (see Figure 6) and grab virtual objects to train them. In our current implementation, we provide *stations* in the environment to train specific physical properties.

Bounciness

Depending on the bounciness $\varepsilon \in [0,1]$ of the object, its motion gets slowed down after hitting a surface and getting reflected. To train the bounciness, users mimic a free fall parabolic motion until it hits the surface followed by a bounce and another parabolic motion. In the following, we describe the calculation of the *coefficient of restitution* (bounciness) for one bounce. Optionally, users can mimic multiple bounces, which yields an averaged coefficient of restitution. The coefficient of restitution for one bounce is defined as follows:

$$\varepsilon = \frac{-\vec{n} \cdot \vec{v}^{+}}{\vec{n} \cdot \vec{v}^{-}} \tag{1}$$

where $\vec{n}$ is the surface normal. $\vec{v}^{-}$ and $\vec{v}^{+}$ are the relative velocities before and after impact.

Instead of measuring the ingoing and outgoing velocity directly, we derive them from the maxima of the trajectory when bouncing on a horizontal surface (i.e., $\vec{n} = (0,1,0)$). This accounts for the fact, that users generally do not describe an accurate parabolic motion with correct timing, but instead think in terms of how high the object bounces. We compare the highest point of the trajectory with respect to the surface before and after the object hits the surface. Figure 3 depicts the vectors involved in the calculation for bounciness. The point p in Figure 3 is the point of impact, p^{-} is the highest point before impact and p^{+} is the highest point after impact.

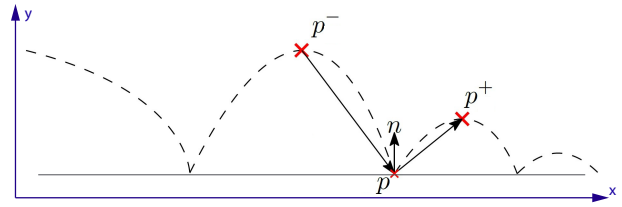

Figure 3. A bouncing motion. The trajectory depicts the x-y-positions of a bouncing object over a time period. We identify the highest points with respect to the surface before and after collision to calculate the bounciness coefficient.

We use the following equations for velocity v and position y with earth's gravity g over time from classical mechanics:

$$v = g \cdot t, \qquad y = \frac{g}{2} \cdot t^2$$

In our case, y is the vertical position of the object. Inserting the formula for y into v yields:

$$v = g \cdot \sqrt{\frac{2 \cdot y}{g}} = \sqrt{2 \cdot y \cdot g} \tag{2}$$

The y-components of p^-, p^+ and p give us two maximum heights.

$$y^-_{max} = p^-_y - p_y, \qquad y^+_{max} = p^+_y - p_y$$

Inserting them into Equation 2 gives us the ingoing and outgoing velocity during impact, respectively. Inserting everything into Equation 1 gives us the formula for bounciness:

$$\varepsilon = \frac{\sqrt{2 \cdot y^+_{max} \cdot g}}{\sqrt{2 \cdot y^-_{max} \cdot g}} = \sqrt{\frac{y^+_{max}}{y^-_{max}}}$$

The highest points of the parabolas are calculated by locally searching for points on the trajectory where the velocity is orthogonal to the surface normal.

Relative mass

To train the mass of an object, users demonstrate the behavior by letting it collide with another object. Figure 2 (b) shows trajectories of two colliding spheres. Depending on how the user changes the velocities of these spheres after collision, their relative mass can be calculated. We assume a non-elastic collision and utilize the *law of conservation of momentum*:

$$M_A \cdot v^-_A + M_B \cdot v^-_B = M_A \cdot v^+_A + M_B \cdot v^+_B$$

Where v^-_A and v^+_A are the velocities of the trained object before and after impact. Same for v^-_B and v^+_B of the other object. M_A is the mass of the object to be trained. M_B is the mass of the other object. Solving for M_A yields:

$$M_A = \frac{M_B \cdot v^+_B - M_B \cdot v^-_B}{v^-_A - v^+_A}$$

M_A can then be used to determine the mass of a third object and so forth. The mass values are unitless and only describe relative masses of objects. The virtual environment is a closed system and therefore objects generally interact correctly as long as users are consistent when demonstrating mass between objects. However, if two objects have not been trained to one another, a default relative mass is assumed, i.e., $M_B = 1$.

Friction

Moving objects that slide along a surface are slowed down depending on the *dynamic friction*. Instead of letting users demonstrate how the objects slow down, we implemented an opposite approach. An object which is sliding down an inclined surface is accelerating. The acceleration depends on the friction and can therefore be used as an indicator for how high the friction is. Users demonstrate the dynamic friction by accelerating an object on an inclined surface (see Figure 5). We base our approach on the *Coulomb model of friction*. Figure 4 depicts the involved vectors. v is the velocity of the object. $m \cdot g$ is the gravitational force pulling the object straight down according to the object's mass. θ is the angle of the inclined surface. We denote the dynamic friction as F. It annihilates the gravitational force depending on the inclination.

$$||F|| = \mu_d \cdot m \cdot g \cdot cos(\theta) \tag{3}$$

Where μ_d is the dynamic friction coefficient we are interested in. We further define the magnitude of the force that makes the object slide down along the surface.

$$||F_s|| = m \cdot g \cdot sin(\theta) \tag{4}$$

The resulting force is then:

$$F_R = F_s + F$$

Furthermore, F_R contains the resulting acceleration.

$$||F_R|| = m \cdot a \tag{5}$$

Note that F_s and F are opposed to each other, hence:

$$||F_R|| = ||F_s|| - ||F|| \tag{6}$$

By inserting equations 3, 4 and 5 into Equation 6, we can solve for the dynamic friction coefficient:

$$m \cdot a = m \cdot g \cdot sin(\theta) - \mu_d \cdot m \cdot g \cdot cos(\theta)$$
$$\mu_d = \frac{g \cdot sin(\theta) - a}{g \cdot cos(\theta)}$$

Users demonstrate how the object constantly accelerates while sliding down the surface. The averaged acceleration of the user's movement gives us a in the equation above.

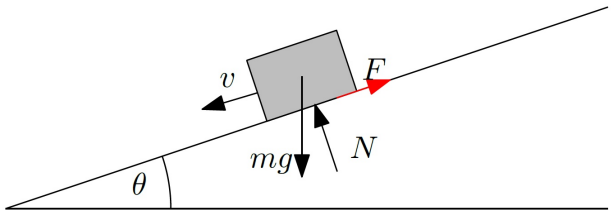

Figure 4. An object sliding down an inclined surface with surface normal N is slowed down by dynamic friction.

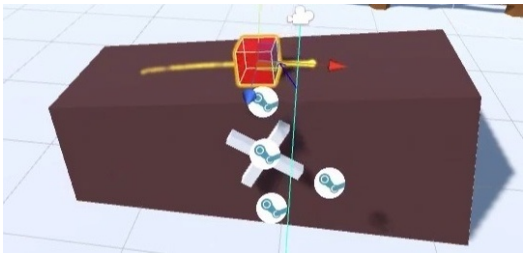

Figure 5. The station for dynamic friction. Users adjust the angle of an inclined surface using the white steering wheel. They can then train dynamic friction by letting a trainable object slide down the surface.

IMPLEMENTATION

Our prototype is implemented in *Unity 5* [1] and is running on an ordinary gaming PC that meets VR requirements. The built-in real-time physics engine of Unity (*PhysX* [14]) allows us to simulate the calculated properties. We use *HTC Vive* and *SteamVR* for our immersive setup. While not being a strict requirement for our approach, we chose to utilize the advantages of VR like leveraging the user's proprioception and using existing 3D interaction techniques. We utilize the *Interaction System* accompanying *SteamVR*. The HTC Vive controllers allow for accurate bimanual 3D trajectory input. Furthermore, they provide physical buttons, which we used to assign functionalities like grasp, start training etc. In order to enable users to navigate in the large virtual environment, we use the built-in Vive teleportation system, which is similar to [8]. Lastly, we have used 3D models from *Low Poly: Free Pack* [5].

CASE STUDY

We conducted a case study to investigate the feasibility of our approach and to gather qualitative feedback on the core idea and the proof-of-concept system with the three implemented properties. We invited 15 participants (6 female) with ages ranging from 21 to 55 (*mean* = 27). Figure 6 shows the virtual environment of our case study. The three stations were introduced one-by-one and participants tried out the functionalities. After the introduction, participants experimented freely and defined properties while expressing their thoughts in a think-aloud protocol. Participants could use the playground to test their defined properties. For instance, they could first train the dynamic friction and mass of an object, go to the playground and try to hit pins by letting it slide over a curved bridge (see Figure 6 bottom). An undo functionality allowed for undoing unintended motions. However, the experimenter was mostly in charge of recovering from errors. A session took around 40 minutes.

Figure 6. Overview of our virtual environment. Virtual objects to train can be obtained from the spawn area. Bounciness, mass and friction can be trained at their respective stations. The playground can be used to test the physical interactions of trained objects.

Results and discussion

We observed that every user got used to the system's capabilities eventually, many of them fairly quickly, even if they did not have much experience in VR. One user commented the following about the overall idea:

> *"The fact that you do it virtual-physical is significantly easier and if I should just set a number between 0 and 1, I would have no clue what did what. [...] Because then you would start with one thing or start with another thing and just try until you hit and could say that oh well, it is approximately that I want. With this method you actually just try it out."*

Another user commented on one of the interactions:

> *I do not understand the physics behind it, but I understand what I must do.*

The majority of users reported enjoyment when using the system. Some users reported that they lost a sense of time, i.e., it took them longer to complete tasks than they perceived. This is in accordance with our goal of seamlessly integrating our approach into possibly playful content creation systems, where efficiency and minimal task completion time are not the foremost objectives, but engagement within the content creation process. It was, however, generally difficult for users to accurately express acceleration compared to velocity or position. Techniques based on our approach should therefore use acceleration carefully.

Limitations and future work

Our approach is targeting novice users. Experienced developers are most likely faster and much more accurate by inputting meaningful numbers in a conventional way. The proof-of-concept system has to deal with technical and human limitations. Examples include inaccuracies like noisy tracking and hand jitter. Furthermore, we are limited by the physical and biomechanical limitations inherent to physically describing curves in space. The performance is also affected by cognitive abilities like timing skills. Our proof-of-concept system implements a very small sub set of possible properties and techniques. Future research could investigate alternative techniques to further explore the design space of our approach. For instance, to train mass, users could adjust the muscle tension of their arm to mimic lifting a heavy object. A different

purpose for our approach would be queries for objects and materials. For instance, by demonstrating how an object slides along a surface could set the virtual surface material to ice.

CONCLUSION

We presented Inpher, a system for setting physical properties using mid-air interaction. The design is based on the ability of humans to express physical motion based on experience and intuition. Our case study showed that users with little physics background were able to train bounciness, relative mass and friction of virtual objects. The results indicate that it is in fact possible to infer physical properties from interactions with virtual objects. Our approach can be used as an integral part of content creation systems for novices. We envision many different applications, which can potentially incorporate our approach. Examples include level editors, puzzle games, interactive systems for education and more.

ACKNOWLEDGMENTS

We would like to thank the case study participants for their valuable feedback. This work has been supported by IFD grant no. 3067-00001B for the project entitled: MADE - A platform for future production.

REFERENCES

1. 2017. Unity 3D. (July 2017). https://unity3d.com/.
2. R. Abraham and J.E. Marsden. 1978. *Foundations of Mechanics*. AMS Chelsea Pub./American Mathematical Society. https://books.google.co.uk/books?id=4Y-ownk6ilsC
3. Priyanshu Agarwal, Suren Kumar, Jason J Corso, and Venkat Krovi. 2011. Estimating dynamics on-the-fly using monocular video. In *Proceedings of 4th Annual Dynamic Systems and Control Conference*.
4. Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.
5. AxeyWorks. 2017. Low Poly: Free Pack. (May 2017). https://www.assetstore.unity3d.com/en/#!/content/58821.
6. Kiran S Bhat, Steven M Seitz, Jovan Popović, and Pradeep K Khosla. 2002. Computing the physical parameters of rigid-body motion from video. In *European Conference on Computer Vision*. Springer, 551–565.
7. Aude G Billard, Sylvain Calinon, and Florent Guenter. 2006. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems* 54, 5 (2006), 370–384.
8. Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. 2016. Point & Teleport Locomotion Technique for Virtual Reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '16)*. ACM, New York, NY, USA, 205–216. DOI: http://dx.doi.org/10.1145/2967934.2968105
9. Andreas Fender, Jörg Müller, and David Lindlbauer. 2015. Creature teacher: A performance-based animation system for creating cyclic movements. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*. ACM, 113–122.
10. HTC. 2017. Vive. (July 2017). https://www.vive.com.
11. Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. 2002. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, Vol. 2. IEEE, 1398–1403.
12. Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. 2000. Interactive control for physically-based animation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 201–208. DOI: http://dx.doi.org/10.1145/344779.344876
13. C Karen Liu, Aaron Hertzmann, and Zoran Popović. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 1071–1081.
14. NVidia. 2017. PhysX. (July 2017). https://developer.nvidia.com/physx-sdk.
15. Antti Oulasvirta, Esko Kurvinen, and Tomi Kankainen. 2003. Understanding contexts by being there: case studies in bodystorming. *Personal and ubiquitous computing* 7, 2 (2003), 125–134.
16. Nancy S Pollard and Jessica K Hodgins. 2004. Generalizing demonstrated manipulation tasks. In *Algorithmic Foundations of Robotics V*. Springer, 523–539.
17. Jovan Popović, Steven M Seitz, and Michael Erdmann. 2003. Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics (TOG)* 22, 4 (2003), 1034–1054.
18. Jovan Popović, Steven M Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 209–217.
19. Jia Sheng, Ravin Balakrishnan, and Karan Singh. 2006. An interface for virtual 3D sculpting via physical proxy.. In *GRAPHITE*, Vol. 6. 213–220.
20. Jia Wang, Owen Leach, and Robert W Lindeman. 2013. DIY World Builder: an immersive level-editing system. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*. IEEE, 195–196.
21. Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. 2016. Physics 101: Learning Physical Object Properties from Unlabeled Videos. In *BMVC*.