ViCMA: Visual Control of Multibody Animations

Doug L. James
djames@cs.stanford.edu
Stanford University
NVIDIA
USA

David I.W. Levin
diwlevin@cs.toronto.edu
University of Toronto
NVIDIA
Canada

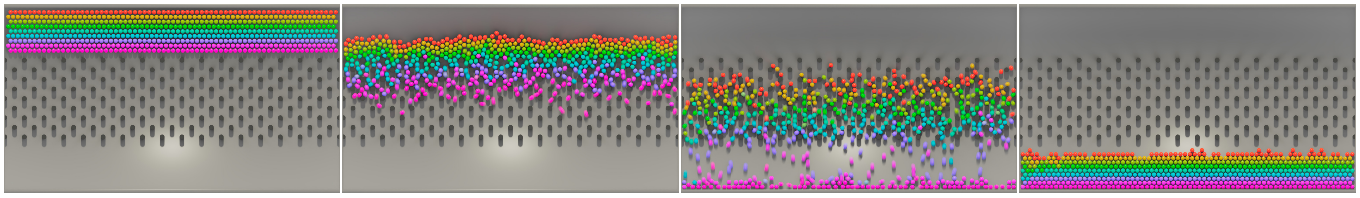

Figure 1: Conservation of Rainbows: ViCMA can convincingly control the initial and final vertical rainbow coloring of 621 balls falling through a Pachinko-style machine. The overall cost is comparable to a single rigid-body simulation.

ABSTRACT

Motion control of large-scale, multibody physics animations with contact is difficult. Existing approaches, such as those based on optimization, are computationally daunting, and, as the number of interacting objects increases, can fail to find satisfactory solutions. We present a new, complementary method for the visual control of multibody animations that exploits object motion and visibility, and has overall cost comparable to a single simulation. Our method is highly practical, and is demonstrated on numerous large-scale, contact-rich examples involving both rigid and deformable bodies.

KEYWORDS

Motion control, animation control, multibody animation, rigid bodies

ACM Reference Format:
Doug L. James and David I.W. Levin. 2023. ViCMA: Visual Control of Multibody Animations. *ACM Trans. Graph.* 1, 1 (September 2023), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Dynamic control of large-scale multibody physics animations is a daunting task. These animations often contain many moving objects in contact-rich scenarios, and their sensitivity to perturbation makes manually or automatically guiding them to an artist-specified outcome practically impossible. These inherent challenges have led to the development of a plethora of experimental control techniques for estimating plausible motions with varying success.

Authors' addresses: Doug L. James, djames@cs.stanford.edu, Stanford University and NVIDIA, USA; David I.W. Levin, diwlevin@cs.toronto.edu, University of Toronto and NVIDIA, Canada.

0730-0301/2023/9-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Optimization-based approaches typically take object end point positions as constraints and attempt to find as-physical-as-possible trajectories between these user specified inputs. However, optimization schemes are computationally expensive and must be tailored to particular simulation or animation techniques. Further, complicated scenes make specifying feasible end-point constraints difficult for the user. The difficulty of achieving success, along with high computational burden, means that optimization techniques for multibody animation control are tedious to apply in practice, especially for many colliding objects.

Alternatively, with browsing and exploration methods, many simulations of a given scene are sampled in parallel, and possibly stored in a compressed, browsable format. A user can execute spatiotemporal queries interactively, in order to locate a particular simulated animation which meets their chosen criteria. Unfortunately, despite large precomputation and storage costs, there is no guarantee that what you want will be sampled, and automated or user-guided search processes can be tedious even for several objects.

Whether optimization or data-driven methods are employed, the practicality of fine-grained animation control of passive multibody systems remains at odds with the sheer difficulty of these problems. Consider historical highlights for the animation control challenge of getting passive rigid bodies to spell phrases: making 30 balls spell "ACM" took 7 days of sampling-based optimization [Chenney and Forsyth 2000]; optimizing one letter-faced die to bounce 6 times in a user-specified fashion so as to spell "SKETCH" took a few minutes [Popović et al. 2003] making 8 letters fall down a Pachinko-style machine and spell "SIGGRAPH" took over an hour of user-browsed parallel rigid-body simulation [Twigg and James 2007]; and making 3037 rigid balls bounce and fall into place to spell a text message took about an hour of compute [Twigg and James 2008]. Given these difficulties, we sought a simpler way to provide control over contact-dominated multibody animations. In contrast, in this work, we spell a text message with 5376 rigid bodies (see SIGGRAPH Card Trick in Figure 2) yet the primary cost is a single simulation, run in minutes. Quantitatively our appearance-based method provides an order of magnitude improvement in scene complexity and performance over previous motion-based methods.

Our insight is that, *instead of using motion control to estimate plausible motions for objects with fixed appearances, it can be more efficient to estimate plausible appearances for objects with fixed motions.* Consequently, our method, ViCMA (Visual Control of Multibody Animations), is simple and practical. It carries out no complicated optimization or sampling, and, in fact, it only requires the animation trajectory to be generated once. The cost of applying the remainder of the algorithm is negligible.

Our approach exploits the fact that multibody animations are often visually chaotic and that the human attention system has difficulty tracking the exact state of objects from frame to frame. We define a heuristic cost function (based on visibility and motion measures) that identifies when a change to an object in the scene is unlikely to be noticed, and executes an appearance transition at a minimum cost frame. This simple and highly practical approach proves efficient and effective for generating a number of enjoyable multibody animations, using significantly less storage and computational resources than alternative methods. Please see Figures 1 and 2 for a preview of our results. Finally, we strongly encourage readers to watch the supplemental video first before reading on and revealing the secrets of ViCMA's magic trick.

2 RELATED WORK

The control of multibody dynamical systems has a long history in computer graphics and animation. Our work is most related to the sub-area of control for *passive* multibody systems (as opposed to *active* control of characters, e.g., using muscles or motor actuators which is beyond the scope of this work). The majority of these works focus on spacetime control of multibody systems, wherein specific spatiotemporal constraints or goals are specified, and control forces or other affordances are optimized to achieve the desired outcomes. Pioneered by Witkin and Kass [1988], spacetime control and its successors have gone on to be extended and applied to many disparate phenomena [Fattal and Lischinski 2004; McNamara et al. 2004; Tang et al. 1995; Thürey et al. 2006; Treuille et al. 2003; Wojtan et al. 2006].

Direct, unsupervised optimization of such systems has its own unique challenges, but particularly difficult are aspects of solid contact, wherein goals may not be physically achievable due to non-penetration constraints. The optimization of motions subject to multibody collision constraints is notoriously difficult for many bodies. As a result, the animation community has focused on various techniques for the estimation of "plausible motion" [Barzel et al. 1996]—in this work, we essentially focus on techniques for estimating *plausible appearances* for given multibody motions. Strategies to overcome motion control difficulties has been to incorporate user feedback either interactively [Cohen 1992; Pan et al. 2013; Popović et al. 2003; Popović et al. 2000; Schoentgen et al. 2020; Yan et al. 2020] or as a precomputed guiding input [Bergou et al. 2007; Forootaninia and Narain 2020; Nielsen and Bridson 2011; Sato et al. 2021; Shi and Yu 2005]. Other control approaches have also been attempted for gentle motions [Barbič and Popović 2008] and multiphysics problems [Ma et al. 2018]. Reverse-time integration has been proposed for end-condition constraints (but then struggles with contact and initial-condition constraints) [Twigg and James 2008], and methods for multiple keyframes on rigid bodies are possible by solving multi-point boundary value problems, but are limited to just a few bodies [Popović et al. 2003]. Solutions for piling phenomena can exploit static analysis to achieve real-time results for tens or hundreds of objects [Hsu and Keyser 2010, 2012] but are not applicable to the dynamic phenomena we study here.

Rather than try to exactly satisfy user constraints, one can instead sample plausible simulation outcomes in search of an acceptable result [Barzel et al. 1996]. Design galleries are an early version of this approach, used to tune simulation parameters [Marks et al. 1997]. Many-Worlds Browsing approaches [Goel and James 2022; Twigg and James 2007] extend this notion by generating a large number of simulation samples and allowing a user to find an acceptable output via interactive spatiotemporal queries and refinement; however, controlling just 8 letters in a Pachinko-like example required significant sampling and user-guided search, whereas ViCMA can visually control hundreds of objects in a Pachinko example with little effort. Data-driven motion graph approaches transition between precomputed rigid or deformable object motions to produce desired control outcomes, which can be challenging given collision constraints [James et al. 2007; Langlois and James 2014].

Chenney and Forsyth [2000] used MCMC sampling to explore the space of rigid-body motions. One notable example involved a die released from a known configuration such that it would fall onto a table, then land in a specifically numbered target circle that also matched the number of pips appearing on the final die orientation (see inset). They optimized both location and final orientation of the die. An alternative approach would be to change the final appearance

All Spots Match
1,000,000 - 1

(or orientation) such that the desired textured number appeared and only sample the location, which would be easier, but it would violate the orientation constraint on the *initial* drop condition. Our work was partly motivated by this alternate approach.

Another option for avoiding the infeasiblity problem is to plausibly relax the physical model such as by including additional noise or perturbations [Barzel et al. 1996]. Amongst other observations (but most pertinent to this work) O'Sullivan and Dingliana [2001] showed that adding distractors to a simulated scene significantly effected observers' ability to detect gaps between colliding objects, and to track collisions in general. O'Sullivan et al. [2003] derived perceptual metrics for evaluating the visual fidelity of physics motion in small scenes, and assessed the important effect that momentum has on collision attention. Reitsma and O'Sullivan [2009] showed that, for scenes involving single digits of rigid bodies that neither variation in appearance, nor audio cues effected a users ability to perceive the realism of a physics simulation. Further work showed that the scenario in which a simulated result is presented effects a users ability to distinguish errors in realism [Reitsma and O'Sullivan 2008]. Based on these perceptual observations, statistical forward simulation has been proposed [Hsu and Keyser 2009], which replaces collision response calculations with precomputed statistical models along with adjusting simulation tolerances based on perceptual metrics [Yeh et al. 2009].

Relevant to ViCMA, O'Sullivan [2005] showed that in scenes involving single digit numbers of colliding rigid bodies, that there

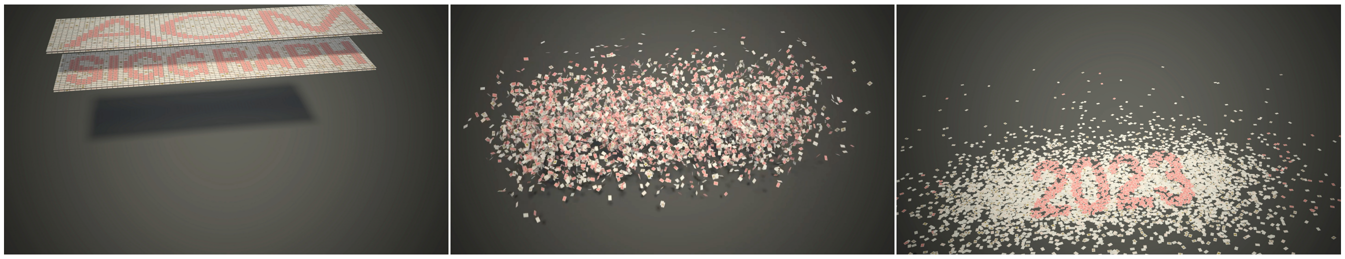

Figure 2: SIGGRAPH Card Trick: ViCMA controls the landing orientation of 5376 textured cards using 2045 texture flips to spell "2023" on the ground. The major runtime cost is a single forward simulation of the falling cards (approx. two minutes in Houdini) and rasterized visibility computations. Since the timing of the texture transitions is computed once, the card message can be changed at a later time (see video for a "Happy Birthday" example).

is a strong attentional effect in collision processing, and Han et al. [2013] showed that humans have difficulty judging simulation accuracy in large-scale simulations. These observations are supported by further work on perception of aggregates [Ramanarayanan et al. 2008]. Taken together these works inspire a different approach altogether: to manipulate object appearance rather than trajectory to give the illusion of keyframe control. We believe our method is the first to exploit visual perception and attention for exclusively appearance-based control of multibody animations.

In particular, our ViCMA optimization is inspired by results of Multiple Object Tracking (MOT) experiments from cognitive science which show that (1) onset of [Abrams and Christ 2003], as well as unexpected [Howard and Holcombe 2010] object motion draws human attention, (2) motion distractors disrupt visual search [Crowe et al. 2021], (3) that motion silences awareness of visual change [Suchow and Alvarez 2011] and that (4) humans have an upper limit on objects they can track simultaneously [Harris et al. 2020b] (shown to be around 4-5). The ViCMA cost heuristic is composed of separate terms that penalize changes for objects at rest (penalizing 1 and exploiting 3), encourages moving distractors and penalizes lone objects (exploiting 2 and 4), coupled with a final visibility term to exploit the well-known phenomena of change blindness [Attwood et al. 2018]. We show, through numerous examples, that this cognitively motivated approach produces compelling illusions of multibody control wherein changing the appearance of more than half the objects in a given scene, in plain sight, goes unnoticed.

3 MULTIBODY APPEARANCE CONTROL

Our approach, Visual Control of Multibody Animations (ViCMA), exploits human inability to reliably attend to large ensembles of objects undergoing complicated interactions [Crowe et al. 2021]. The input to our algorithm is a set of keyframes containing prescribed appearance properties (e.g., color, texture, etc.) for all (or some) objects in the animation. Rather than globally (or even locally) optimize object trajectories, we instead transition object appearance attributes to create the appearance of a key-framed, trajectory-controlled optimization.

To do this, our algorithm computes a transition cost (on $[0, 1]$) for each object in a scene, at each time frame. Crucially, this cost computation is local, depending only on the instantaneous state (position and velocity) of a given object and a small number of spatial neighbors. This local dependence enables fast cost computation on even large examples. Eschewing more involved optimization procedures, we transition an object's appearance attributes at its individual minimum cost time, which can be computed quickly by scanning over all per-object, per-frame transition costs.

3.1 Visibility Cost

When possible we exploit visibility changes to transition objects when they are out of sight, or have low projected area in the frame. We measure per-frame visibility as the projected area of an object on the scene which we compute from rasterized object ID images (Figure 5). For an object, i, we compute

$$\phi_i^{viz} = \frac{1}{N_{pixels}} \sum_{j,k=1}^{N_{pixels}} (\mathrm{ID}_i == \mathrm{ID}\,(j,k)), \tag{1}$$

where N_{pixels} is the number of pixels in the ID image, ID_i is the assigned identification value for the i^{th} object and $\mathrm{ID}\,(j,k)$ is the identification value stores at pixel (j,k). The logical operator == evaluates to 1 if IDs are equal, and 0 otherwise.

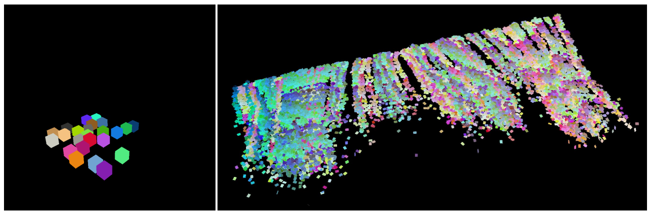

Figure 5: Rasterization-based Visibility Calculations: RGBA ID images from (Left) the Yahtzee, and (Right) the Faulty Card Towers animations.

3.2 Transition Cost for Many-body Motion

While visibility can be exploited to determine when to change appearances, challenging scenarios exist where the objects are always visible, e.g., rainbow-colored balls falling through a pachinko-like machine (see Figure 1). In such cases, we must exploit the chaotic nature of the motion and mixing to hide appearance changes. In this section, we will describe some cost measures that we experimentally found useful for modeling appearance transitions in practice.

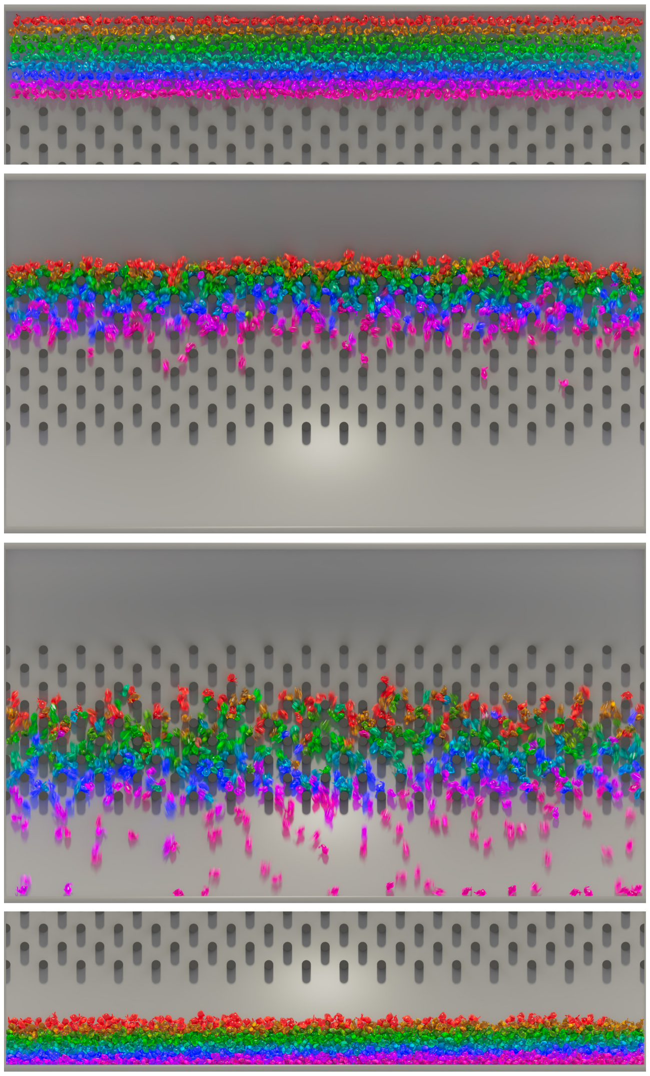
Figure 3: "Bunchinko!" We applied ViCMA to deformable bodies as demonstrated by this rainbow-to-rainbow pachinko machine using 621 Stanford gummy bunnies (simulated in Houdini Vellum (256 tri, 130 vtx)). This example is challenging not only because objects are always visible, but also because the deformable motions cause bunnies to squish upon peg impact, then get piled-up upon and passed by, which changes color ordering noticeably. The motion cost function, ϕ^{motion}, is computed using center-of-mass velocity and position attributes. (ViCMA Score of 0.570)

Change during movement. Our first velocity-level cost measure penalizes changes for objects at rest. Let the velocity of particle/body i be $\mathbf{v}_i$, with speed $v_i = \|\mathbf{v}_i\|$. A simple cost function based on single-body velocity is

$$\phi_i^v = \exp\left[-\left(\frac{v_i}{\delta v}\right)^2\right] \tag{2}$$

where δv is a user-defined scalar parameter. This measure allows a single fast-moving body transition, but the transition would be less apparent if surrounding objects were also moving in an incoherent way. To measure the latter, we measure the velocity standard deviation, σ_i, of the surrounding objects:

$$\sigma_i^2 = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\|\bar{\mathbf{v}}_i - \mathbf{v}_j\|^2 \quad \text{with} \quad \bar{\mathbf{v}}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\mathbf{v}_j, \tag{3}$$

where $\mathcal{N}_i$ is a list of local neighbors (excluding i) with centroids within a user-defined threshold, R; in our examples, we use $R = 4r$ where r is the object bounding radius, unless mentioned otherwise. If $\mathcal{N}_i$ is empty, we set $\sigma_i = 0$. Note that σ_i also requires at least two neighbors to be nonzero.

Our velocity-level cost term for object i at the current frame is

$$\phi_i^{vel} = \exp\left[-\left(\frac{\min(v_i,\sigma_i)}{\delta v}\right)^2\right] \in [0,1], \tag{4}$$

where the min operation ensures the cost is high if the object is not moving sufficiently, or the neighbors are not moving incoherently enough. In practice, ϕ_i^{vel} does a good job of finding locations where the object is moving, and at least two neighbors are moving disparately. Figure 6 (top row) shows its temporal evolution.

Encouraging moving diversions. An object's velocity-level cost is low if it is moving, and has disparately moving neighbors (Figure 6:top row). However, a lone fast-moving object can impact another non-moving object or objects, imparting velocity to the latter. Consequently, this velocity-level cost alone can create transitions where a single fast-moving object hits stationary objects and then changes its color in an obvious manner. To avoid objects changing their color when they stand out, we add an additional cost term that encourages more moving neighbors by explicitly counting them. We count the number of neighbors in $\mathcal{N}_i$ with sufficient speed, $v_j > \delta v$. Also, we avoid pure translational motion cases (which aren't good diversions) by requiring the neighbors' velocities $\mathbf{v}_j$ to be different from $\mathbf{v}_i$. Let the number of moving local neighbors be

$$n_i^{mov} = \sum_{j\in\mathcal{N}_i}\left(\min(v_i, \|\mathbf{v}_i - \mathbf{v}_j\|) > \delta v\right) \tag{5}$$

where the boolean true/false comparison (>) evaluates to 1/0. Finally, the diversion cost uses a cubic smoothstep to encourage more than n_{min} moving neighbors:

$$\phi_i^{mov} = 1 - \text{smooth}(n_{min}, n_{max}, n_i^{mov}). \tag{6}$$

In our examples we request more than three moving neighbors ($n_{min} = 3$, $n_{max} = 5$) to encourage appearance-change diversions when possible. The second row of Figure 6 shows the temporal evolution of ϕ^{mov}.

Wait for mixing. Early impacts and shock waves can create multi-body velocity disturbances that have both low velocity-level and moving diversion costs, ϕ^{vel} and ϕ^{mov}, respectively (Figure 6: second row). These noisy velocity-level measures alone are therefore not robust to collision disturbances, and can cause a closely packed arrangement of objects, e.g., a grid of pachinko balls, to suddenly and obviously change their colors before much movement has even occurred. We therefore also discourage transitions from occurring before sufficient spatial mixing has occurred, which is more robust

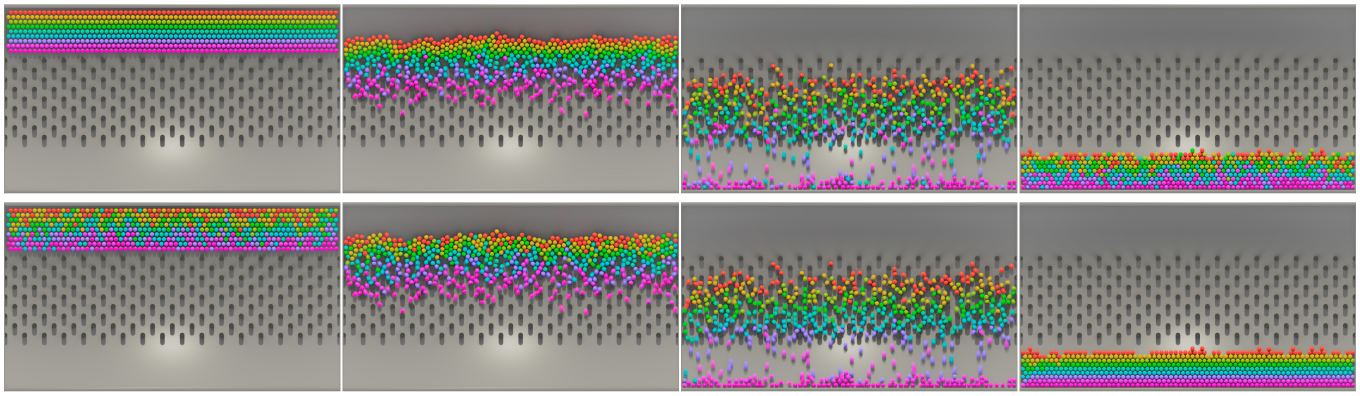

Figure 4: Prior methods do not conserve rainbows: Initial vs final color constraints applied to the same Pachinko simulation from Figure 1. Specifying ball colors with (Top) an initial rainbow color condition produces an animation which runs forward and gets mixed up, whereas (Bottom) a final rainbow color condition sets colors which start out mixed up, but then run forward in time to satisfy the final rainbow constraint (e.g., see popular work by Konstantin Otrembsky [Dean 2018]). In contrast, ViCMA can support both initial and final color constraints, and estimate plausible color transitions for the many balls which will have infeasible (non-matching) two-point color conditions.

than just using velocity-level measures of change. Historically, using spatial rearrangement can confuse a viewer, such as during a "shell game" [Wikipedia 2004]. In our case, the sleight-of-hand color switch occurs in plain sight, which is particularly brazen.

We now discuss how to measure the similarity of an object's local neighborhood at the current frame (t) to, e.g., the beginning frame ($t = 0$). Given the radius-R neighbors of object i at the start, $\mathcal{N}_i^0$, and at a later time, $\mathcal{N}_i^t$, when are these objects sufficiently different so that we can allow transitions? We considered discrete set-similarity measures [Vijaymeena and Kavitha 2016]: for example, given two neighborhood sets, $\mathcal{A}$ and $\mathcal{B}$, the Jaccard index is the ratio of the intersection to union sizes, $|\mathcal{A} \cap \mathcal{B}|/|\mathcal{A} \cup \mathcal{B}| \in [0, 1]$, whereas the overlap (or Szymkiewicz–Simpson) coefficient is $|\mathcal{A} \cap \mathcal{B}|/\min(|\mathcal{A}|,|\mathcal{B}|) \in [0, 1]$. We found the latter most useful for ruling out early transitions, but too conservative, and it would sometimes indicate that the neighborhoods were similar well into the simulation when neighbors were still nearby but had rearranged spatially.

Therefore, we devised a *directional overlap coefficient* normalized by the reference (start or end) neighborhood set size. Instead of just adding one for each member j of the intersecting neighborhood sets, e.g., $j \in \mathcal{N}_i^0 \cap \mathcal{N}_i^t$, we accumulate the dot product of the normalized displacement vectors from object location i to neighbor j between the two time frames (clamped to $[0, 1]$). Specifically, we define the *directional overlap coefficient* between the starting (0) and current (t) frame as

$$D_i^{0t} = \frac{1}{|\mathcal{N}_i^0|} \sum_{j \in \mathcal{N}_i^0 \cap \mathcal{N}_i^t} \left(\hat{\mathbf{d}}_{ji}^0 \cdot \hat{\mathbf{d}}_{ji}^t \right)_+, \quad |\mathcal{N}_i^0| > 0, \tag{7}$$

and zero for isolated objects with $\mathcal{N}_i^0 = \emptyset$; here $\hat{\mathbf{d}}_{ji}$ are the normalized displacement vectors from i to j

$$\hat{\mathbf{d}}_{ji} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \tag{8}$$

evaluated at the appropriate time (here 0 or t). We also apply this to end-frame matching by replacing "0" by the end frame index, F, to get end-current directional overlap coefficient, D_i^{Ft}.

Finally, our overall neighborhood mixing cost at frame t is the max of both costs:

$$\phi_i^{mix} = \max(D_i^{0t}, D_i^{Ft}). \tag{9}$$

This cost is always 1 at beginning (0) and end frames (F), and zero for particles that are isolated at t or their reference frame (0 or F) (Figure 6: third row).

3.3 Combined Cost Model for Multibody Animation

It is possible to sum all cost functions together, however to strongly discourage unmixed transitions, we use the following motion cost model for body i:

$$\phi_i^{motion} = \max\left(\phi_i^{mix}, \frac{1}{2}\left(\phi_i^{vel} + \phi_i^{mov}\right)\right). \tag{10}$$

In other words, we use the average of the two velocity-level cost functions unless the neighborhood is insufficiently mixed. Note that the multibody motion cost is one for an isolated body.

In practice, a simulation might have a lone visible object for which $\phi^{motion} = 1$ throughout the animation. In cases where we must rely on velocity to determine a transition (and not visibility or other factors), we can regularize the motion cost by adding a tiny single-body velocity cost (2),

$$\phi_i^{motion} = \max\left(\phi_i^{mix}, \frac{1}{2}\left(\phi_i^{vel} + \phi_i^{mov}\right)\right) + \varepsilon_v \, \phi_i^{v}, \tag{11}$$

where we use $\varepsilon_v = 0.001$ in our examples (Figure 6: fourth row).

Lastly, we multiply motion and visibility costs to arrive at the final per-frame ViCMA cost for body i:

$$\phi_i^{vicma} = \phi_i^{motion} \, \phi_i^{viz}. \tag{12}$$

Valid transition times are minimizers of ϕ_i^{vicma}, and are computed once and cached. Figure 6 shows all costs evaluated at multiple frames for a single animation.

3.4 Scoring Animations

Some animations and constraints are better suited to ViCMA than others. We can report a per-body transition cost based on the min-cost achieved (Figure 6: bottom), weighted by other factors, such as whether or not a transition was needed (0 or 1), color change amount, etc. We define the *ViCMA Score* as the maximum per-body transition cost observed (lower is better). In practice, the animator can try a few animations and pick the one with the lowest ViCMA Score. Most of our examples were run through ViCMA "as is" with the majority of time spent tuning physics simulation parameters.

4 RESULTS

ViCMA has been used to control numerous large multibody animations. We strongly encourage readers to enjoy the supplemental ViCMA animation results before continuing. ViCMA parameters were the same for all examples (mentioned previously), but, because ViCMA is fast, it is possible to tune parameters as desired.

Implementation and Performance: ViCMA is easy to implement. The solver is local in object and time, iterating over each object in parallel and selecting the transition time as the first, minimum cost frame. We used two implementations to generate the results for this paper, one for Houdini [SideFX 2023] and another for the open source tool Blender [2023]. The Blender implementation is written in Python using scikit-learn [Pedregosa et al. 2011] for nearest neighbour queries and numpy [Harris et al. 2020a] for algebraic operations. By far the slowest component of ViCMA is creating an initial multibody animation (in this paper done using rigid and deformable physics simulation). Rasterizing object ID images, which we use for visibility computation, is embarrassingly parallel and performed on CPU (Houdini) or GPU (Blender). In practice, computing the ViCMA cost and finding minimum transition frames takes less than 5 seconds on an Intel Core i9 workstation with 32GB of RAM. No effort was made to optimize the code beyond what is provided automatically by the underlying tools. Even so, we were able to generate controlled multibody animations on scenes with thousands of colliding objects.

Facial Expressions (rigid, colors): Figure 7 shows the result of keyframing object color to alter the facial expression mapped onto an animation of 2500 falling rigid cubes. Despite being a very short animation with a single main impact period, ViCMA identifies times when objects are occluded or undergo sufficient motion and mixing to transition colors on 591 cubes in a plausible manner.

Pachinko (rigid, colors): Shown in Figure 1, this example is challenging because objects are visible at all times during the animation. The 623 balls in this scene initially form a rainbow, and ViCMA find 406 transitions to hit a rainbow-structured keyframe at the end of the animation in a believable manner. Initial velocity noise is introduced for plausibility.

Unchinko (rigid, colors): By removing all Pachinko pegs, we can produce a visually much simpler falling motion. This scenario is similar to the falling-cubes "Facial Expression" example, but more difficult because it is 2D and lacks visibility-change affordances. Surprisingly, ViCMA is able to find visually appropriate times at which to perform color transitions (Figure 8) during the rapid and visually chaotic impact phase.

Padrinko (rigid, colors): Because ViCMA is a post-process, applied to an existing multibody animation, it is readily compatible with complex, fast-moving scenes. In Figure 9, ViCMA finds 543 transitions to distribute 621 rapidly moving multi-colored balls into 9 independent bins ("drink cups") which raise from the ground. Interestingly, each cup is filled with a vertical rainbow even though the original ball rainbow was horizontal. While motion control would be essentially impossible using previous techniques, yet ViCMA takes only seconds to compute, and achieves our lowest ViCMA Score of 0.0372 due to the fast-moving chaotic motion.

Bunchinko (deformable, colors): ViCMA is simulator agnostic and so creating animations with deformable objects is trivial. This example replaces the 621 rigid Pachinko balls with deformable Stanford Bunnies (see Figure 3). ViCMA preserves the starting and ending rainbow structure using 445 transitions.

Cost Function Analysis and Ablation Study: The video visualizes the cost functions from Figure 6 for the Pachinko example (Figure 1). It also includes an ablation study that demonstrates the animations resulting from the progressively better sequence of cost models: (1) first just the single-body velocity cost, ϕ^v, which transitions when the body moves the fastest and produces unconvincing results; (2) then the multibody velocity cost, $(\phi^{vel}+\phi^{mov})/2$, which transitions when neighbors are moving more chaotically, but can also have spurious and sometimes obvious transitions in unmixed early/late states; and, finally, (3) the full ViCMA motion cost ϕ^{motion} which incorporates the mixing cost ϕ^{mix} to transition during more confusing motions and avoid unmixed early/late transitions. We also plot transition times as obvious red-to-green color transitions for clarity. These results parallel discussions in the cost section.

B#ggle Dice (rigid, textures): ViCMA can be used to transition other appearance properties such as textures. Figure 10 appears to show an example of pose control, however for examples such as dice rolls, we can imitate pose control via dynamic texturing. Here ViCMA transitions textures 23 out of 25 dice in order to give the appearance of an impossible dice roll: from all fours showing to, all sixes showing (a one in $6^{25} \approx 3 \times 10^{19}$ chance). This animation is challenging because the dice are in full view for most of the sequence. ViCMA locates transition times during which objects are undergoing sufficient movement and mixing to transition seamlessly.

Dice Pyramid Explosion (rigid, textures): Texture transition effects can be applied to large-scale simulations. In Figure 11, a dice pyramid consisting of 5456 rigid bodies is exploded. All the dice move from an initial structured arrangement with sixes up, to landing sixes-up at the end of the animation. ViCMA is able to exploit visibility and motion to find unnoticeable texture transitions for all objects in the scene (526 transitions, ignoring initial hidden-inside-pile dice). The mixing cost function was key to avoiding spurious settling dice from switching textures close to the camera.

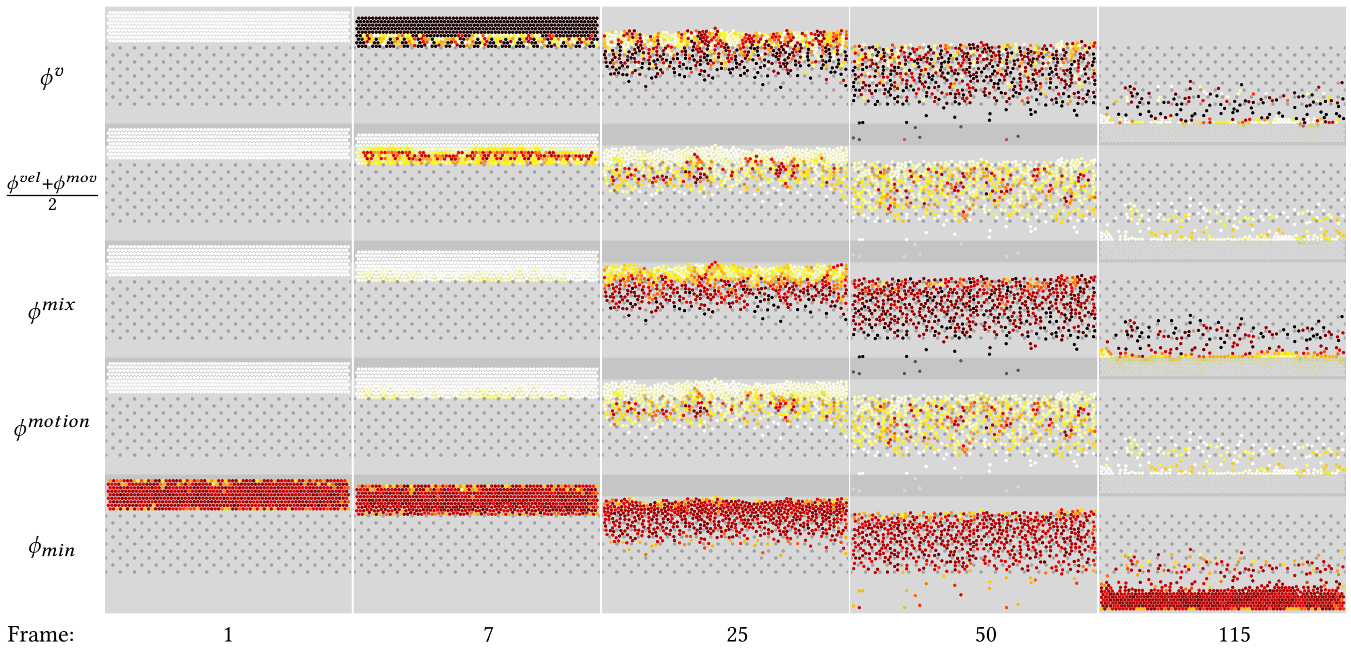

Figure 6: Motion Cost Analysis: (Top to Bottom) The single-body velocity cost, ϕ^{v}, is very noisy, and even wants to transition during the shockwave on frame 7; the neighbor-based multibody velocity costs, $\frac{\phi^{vel}+\phi^{mov}}{2}$, identify nearby bodies with disparate velocities; the direction overlap coefficients in $\phi^{mix} = \max(D^{0t}, D^{Ft})$ have high costs near start/end neighborhood configurations; the overall multibody motion cost, ϕ^{motion}; and the min-cost over all frames. The original animation consists of 180 frames. Normalized costs plotted using a black-body color map: 0 1.

Faulty Card Towers (rigid, colors, visibility only): We built bi-colored card towers with regular color patterns and blew them down onto the ground to create a bold stripe pattern (Figure 12). In this example, we only used the visibility costs to make the 16190 falling cards undergo 8076 transitions when they were (approximately) hidden. The card simulation was done using Bullet in Houdini, and was the dominant cost.

ACM SIGGRAPH Card Trick (rigid, textured): In this large example involving 5376 double-sided textured cards (see Figure 2) cards initially spelling "ACM SIGGRAPH" then fall down to reveal "2023" on the ground. The light fluttering motion of the cards was achieved using a nonlinear aerodynamic lift model for falling paper inspired by [Pesavento and Wang 2004]. The falling and tumbling motion provides many opportunities for low-cost appearance change since the planar cards can have many low cross-sectional-area moments with small visibility costs. The highest card costs tend to arise for cards that somehow manage to fall without tumbling, remain facing the camera, and avoid significant occlusion. See the supplemental video for a visual analysis of costs, transitions, and view-dependent nature of simulated card transitions (shown as yellow flashes).

5 CONCLUSION

We explored a number of techniques, based on exploiting visibility and motion measures, for the visual control of multibody animations (ViCMA). We have shown that ViCMA can produce compelling visual effects when applied to rigid and deformable animations, animations which involve copious amounts of contact, and animations that require dynamic texture mapping. We have been able to demonstrate ViCMA on very large examples because of our focus on simple and efficient appearance-based techniques: ViCMA requires no complicated spacetime optimization, it runs in seconds even for our largest examples, and precomputed cost maps can be reused, e.g., to change colors. We believe these techniques are eminently practical and complementary to prior techniques for spacetime motion control—we are not aware of prior motion control techniques that can achieve comparable results. Perhaps the most exciting part of ViCMA is that it introduces the notion of exploiting human attention for animation control to computer graphics, and that we have only scratched the surface of how to apply "visual control" to computer animation.

Limitations and Discussion. ViCMA cost functions ignore rotational motion which can be important when perceiving textures, e.g., spinning dice. Visibility-based costs are view dependent, and the resulting illusion can degrade by looking at the animation from a different viewpoint or via material reflections (see video for an illustration). There is no guarantee that a visible transition will go unnoticed by an attentive viewer. Color changes may be less noticeable in peripheral vision, but overall brightness changes less so. ViCMA can produce compelling results in some cases, but it is not well-suited for all animation tasks. Animations with few or isolated objects, or coherent motions, often don't admit low-cost transitions. Keyframes need to have similar color distributions or appearances otherwise no subtle transitions can be found. Some animation effects inherently require trajectory modification, which

ViCMA cannot perform by construction, and traditional motion control techniques may produce more pleasing results with less visible artifacts. Future work should find ways to use ViCMA in a complementary role, along side space-time optimization, sampling and browsing techniques. Finally, we leave generalizing to objects with different physical sizes and textures for future work. Color and texture were not considered by construction in order to allow prerendering of visibility frames and so choosing wildly disparate start and end colors or texture will produce noticeable transitions.

ViCMA's efficiency comes from the fact that the cost at each time step is independent of object state at other time steps, and that bodies can be optimized in parallel. This design choice allows ViCMA to be applied to large-scale simulations, and allows transitions to be reused with different appearance data. However, it means passing up on further opportunities to optimize color and texture transitions (such as taking into account neighboring color information, or reasoning about when a lone color might stand out). The exploration of coupled-body optimization procedures remains future work.

Acknowledgements. The authors thank the reviewers for their constructive suggestions; Ken Museth and the High-fidelity Physics Simulations for Omniverse group for interesting conversations and helpful guidance; and software from Blender, SideFX and Adobe.

REFERENCES

Richard A. Abrams and Shawn E. Christ. 2003. Motion Onset Captures Attention. *Psychological Science* 14, 5 (2003), 427–432. https://doi.org/10.1111/1467-9280.01458 arXiv:https://doi.org/10.1111/1467-9280.01458 PMID: 12930472.

Jonathan E Attwood, Christopher Kennard, Jim Harris, Glyn Humphreys, and Chrystalina A Antoniades. 2018. A comparison of change blindness in real-world and on-screen viewing of museum artefacts. *Frontiers in Psychology* 9 (2018), 151.

Jernej Barbič and Jovan Popović. 2008. Real-Time Control of Physically Based Simulations Using Gentle Forces. *ACM Trans. Graph.* 27, 5, Article 163 (dec 2008), 10 pages. https://doi.org/10.1145/1409060.1409116

Ronen Barzel, John R Hughes, and Daniel N Wood. 1996. Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation'96*. Springer, 183–197.

Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: Toward Directable Thin Shells. *ACM Trans. Graph.* 26, 3 (jul 2007), 50–es. https://doi.org/10.1145/1276377.1276439

Blender. 2023. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

Stephen Chenney and D. A. Forsyth. 2000. Sampling Plausible Solutions to Multi-Body Constraint Problems. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 219–228. https://doi.org/10.1145/344779.344882

Michael F. Cohen. 1992. Interactive Spacetime Control for Animation. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*. Association for Computing Machinery, New York, NY, USA, 293–302. https://doi.org/10.1145/133994.134083

Emily M Crowe, Christina J Howard, Iain D Gilchrist, and Christopher Kent. 2021. Motion disrupts dynamic visual search for an orientation change. *Cognitive research: principles and implications* 6, 1 (2021), 1–9.

Signe Dean. 2018. This Machine "Quantum" Sorting Marbles Into a Rainbow Actually Has a Dirty Secret. Science Alert. https://www.sciencealert.com/coloured-balls-sorted-machine-quantum-resonance-fake-computer-simulation-galton-board

Raanan Fattal and Dani Lischinski. 2004. Target-driven Smoke Animation. In *ACM SIGGRAPH 2004 Papers*. 441–448.

Zahra Forootaninia and Rahul Narain. 2020. Frequency-Domain Smoke Guiding. *ACM Trans. Graph.* 39, 6, Article 172 (nov 2020), 10 pages. https://doi.org/10.1145/3414685.3417842

Purvi Goel and Doug L. James. 2022. Unified Many-Worlds Browsing of Arbitrary Physics-Based Animations. *ACM Trans. Graph.* 41, 4, Article 156 (jul 2022), 15 pages. https://doi.org/10.1145/3528223.3530082

Donghui Han, Shu-wei Hsu, Ann McNamara, and John Keyser. 2013. Believability in Simplifications of Large Scale Physically Based Simulation. In *Proceedings of the ACM Symposium on Applied Perception* (Dublin, Ireland) *(SAP '13)*. Association for Computing Machinery, New York, NY, USA, 99–106. https://doi.org/10.1145/2492494.2492504

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020a. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362.

David J Harris, Mark R Wilson, Emily M Crowe, and Samuel J Vine. 2020b. Examining the roles of working memory and visual attention in multiple object tracking expertise. *Cognitive processing* 21 (2020), 209–222.

Christina J Howard and Alex O Holcombe. 2010. Unexpected changes in direction of motion attract attention. *Attention, Perception, & Psychophysics* 72, 8 (2010), 2087–2095.

Shu-Wei Hsu and John Keyser. 2009. Statistical Simulation of Rigid Bodies. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New Orleans, Louisiana) *(SCA '09)*. Association for Computing Machinery, New York, NY, USA, 139–148. https://doi.org/10.1145/1599470.1599489

Shu-Wei Hsu and John Keyser. 2010. Piles of Objects. *ACM Trans. Graph.* 29, 6, Article 155 (dec 2010), 6 pages. https://doi.org/10.1145/1882261.1866181

Shu-Wei Hsu and John Keyser. 2012. Automated Constraint Placement to Maintain Pile Shape. *ACM Trans. Graph.* 31, 6, Article 150 (nov 2012), 6 pages. https://doi.org/10.1145/2366145.2366169

Doug L James, Christopher D Twigg, Andrew Cove, and Robert Y Wang. 2007. Mesh Ensemble Motion Graphs: Data-driven mesh animation with constraints. *ACM Transactions on Graphics (TOG)* 26, 4 (2007), 17–es.

Timothy R. Langlois and Doug L. James. 2014. Inverse-Foley Animation: Synchronizing rigid-body motions to sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33, 4 (Aug. 2014). https://doi.org/10.1145/2601097.2601178

Pingchuan Ma, Yunsheng Tian, Zherong Pan, Bo Ren, and Dinesh Manocha. 2018. Fluid Directed Rigid Body Control Using Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 4, Article 96 (July 2018), 11 pages. https://doi.org/10.1145/3197517.3201334

J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 389–400. https://doi.org/10.1145/258734.258887

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid Control Using the Adjoint Method. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 449–456. https://doi.org/10.1145/1015706.1015744

Michael B. Nielsen and Robert Bridson. 2011. Guide Shapes for High Resolution Naturalistic Liquid Simulation. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 83, 8 pages. https://doi.org/10.1145/1964921.1964978

Carol O'Sullivan. 2005. Collisions and Attention. *ACM Trans. Appl. Percept.* 2, 3 (jul 2005), 309–321. https://doi.org/10.1145/1077399.1077407

Carol O'Sullivan and John Dingliana. 2001. Collisions and Perception. *ACM Trans. Graph.* 20, 3 (jul 2001), 151–168. https://doi.org/10.1145/501786.501788

Carol O'Sullivan, John Dingliana, Thanh Giang, and Mary K. Kaiser. 2003. Evaluating the Visual Fidelity of Physically Based Animations. In *ACM SIGGRAPH 2003 Papers* (San Diego, California) *(SIGGRAPH '03)*. Association for Computing Machinery, New York, NY, USA, 527–536. https://doi.org/10.1145/1201775.882303

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive Localized Liquid Motion Editing. *ACM Trans. Graph.* 32, 6, Article 184 (Nov. 2013), 10 pages. https://doi.org/10.1145/2508363.2508429

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

Umberto Pesavento and Z Jane Wang. 2004. Falling Paper: Navier-Stokes solutions, model of fluid forces, and center of mass elevation. *Physical review letters* 93, 14 (2004), 144501.

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.

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 (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–217. https://doi.org/10.1145/344779.344880

Ganesh Ramanarayanan, Kavita Bala, and James A Ferwerda. 2008. Perception of complex aggregates. In *ACM SIGGRAPH 2008 papers*. 1–10.

Paul S. A. Reitsma and Carol O'Sullivan. 2008. Effect of Scenario on Perceptual Sensitivity to Errors in Animation. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization* (Los Angeles, California) *(APGV*

'08). Association for Computing Machinery, New York, NY, USA, 115–121. https://doi.org/10.1145/1394281.1394302

Paul S. A. Reitsma and Carol O'Sullivan. 2009. Effect of Scenario on Perceptual Sensitivity to Errors in Animation. *ACM Trans. Appl. Percept.* 6, 3, Article 15 (sep 2009), 16 pages. https://doi.org/10.1145/1577755.1577758

Syuhei Sato, Yoshinori Dobashi, and Theodore Kim. 2021. Stream-Guided Smoke Simulations. *ACM Trans. Graph.* 40, 4, Article 161 (July 2021), 7 pages. https://doi.org/10.1145/3450626.3459846

Arnaud Schoentgen, Pierre Poulin, Emmanuelle Darles, and Philippe Meseure. 2020. *Particle-Based Liquid Control Using Animation Templates*. Eurographics Association, Goslar, DEU. https://doi.org/10.1111/cgf.14103

Lin Shi and Yizhou Yu. 2005. Controllable Smoke Animation with Guiding Objects. *ACM Transactions on Graphics* 24 (01 2005). https://doi.org/10.1145/1037957.1037965

SideFX. 2023. Houdini Engine. http://www.sidefx.com.

Jordan W Suchow and George A Alvarez. 2011. Motion silences awareness of visual change. *Current Biology* 21, 2 (2011), 140–143.

Diane Tang, J Thomas Ngo, and Joe Marks. 1995. N-body Spacetime Constraints. *The Journal of Visualization and Computer Animation* 6, 3 (1995), 143–154.

N. Thürey, R. Keiser, M. Pauly, and U. Rüde. 2006. Detail-Preserving Fluid Control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vienna, Austria) *(SCA '06)*. Eurographics Association, Goslar, DEU, 7–12.

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe Control of Smoke Simulations. *ACM Trans. Graph.* 22, 3 (jul 2003), 716–723. https://doi.org/10.1145/882262.882337

Christopher D. Twigg and Doug L. James. 2007. Many-Worlds Browsing for Control of Multibody Dynamics. *ACM Trans. Graph.* 26, 3 (July 2007), 14–es. https://doi.org/10.1145/1276377.1276395

Christopher D Twigg and Doug L James. 2008. Backward steps in rigid body simulation. In *ACM SIGGRAPH 2008 papers*. 1–10.

MK Vijaymeena and K Kavitha. 2016. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal* 3, 2 (2016), 19–28.

Wikipedia. 2004. Shell Game — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Shell_game [Online; accessed 22-August-2023].

Andrew Witkin and Michael Kass. 1988. Spacetime Constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '88)*. Association for Computing Machinery, New York, NY, USA, 159–168. https://doi.org/10.1145/54852.378507

Chris Wojtan, Peter J Mucha, and Greg Turk. 2006. Keyframe control of complex particle systems using the adjoint method. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 15–23.

Guowei Yan, Zhili Chen, Jimei Yang, and Huamin Wang. 2020. Interactive Liquid Splash Modeling by User Sketches. *ACM Trans. Graph.* 39, 6, Article 165 (Nov. 2020), 13 pages. https://doi.org/10.1145/3414685.3417832

Thomas Y. Yeh, Glenn Reinman, Sanjay J. Patel, and Petros Faloutsos. 2009. Fool Me Twice: Exploring and Exploiting Error Tolerance in Physics-Based Animation. *ACM Trans. Graph.* 29, 1, Article 5 (dec 2009), 11 pages. https://doi.org/10.1145/1640443.1640448

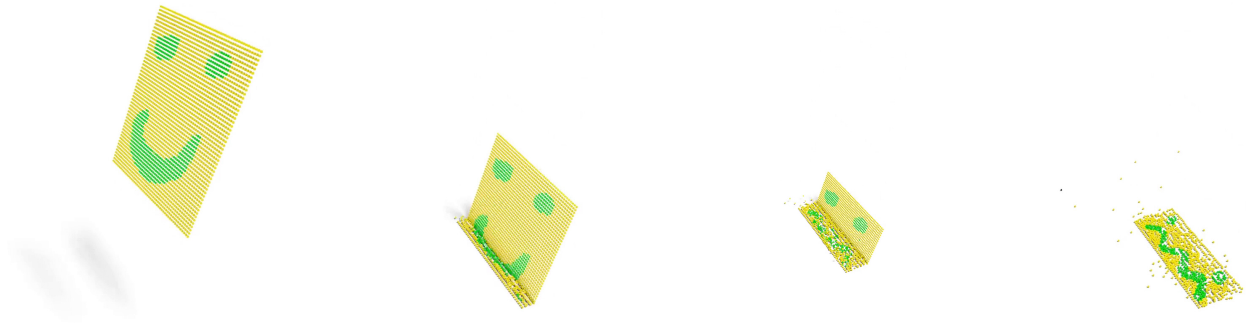
Figure 7: Keyframed Expression: We constrain the start and end facial expressions of this simulation of 2500 cubes falling on the ground (591 transitions).

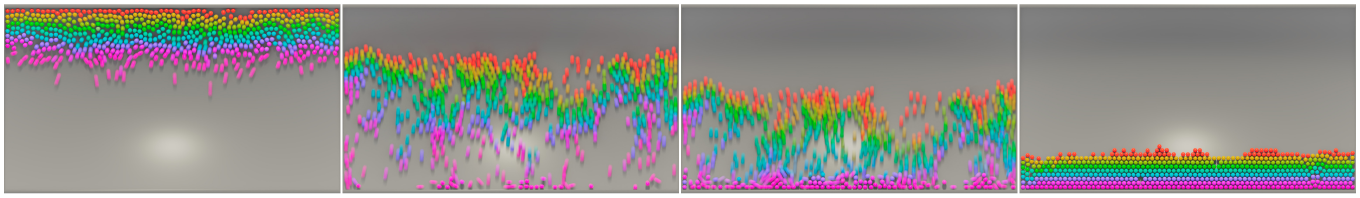
Figure 8: Dropping the Balls (Unchinko): These balls have randomized initial velocities and fall through an empty box, and land in a rainbow arrangement. Despite ViCMA transitioning the colors of 425 out of 621 balls, and mostly when the objects hit the ground, the changes are surprisingly plausible (ViCMA Score of 0.758).

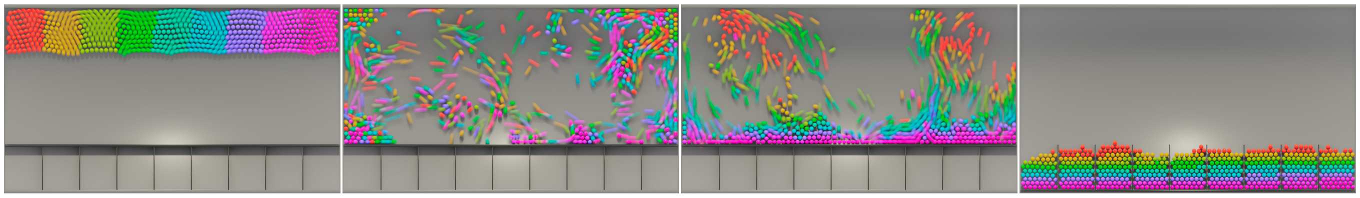
Figure 9: Re-sorting Colors (Padrinko): Horizontally sorted colors are rapidly mixed using noise-based forces, and magically fall into a vertically sorted rainbow color pattern, which are lowered into bins. The violent mixing process provides many extremely low-cost opportunities for ViCMA to color-transition 543 out of 621 balls (ViCMA Score of 0.0372).

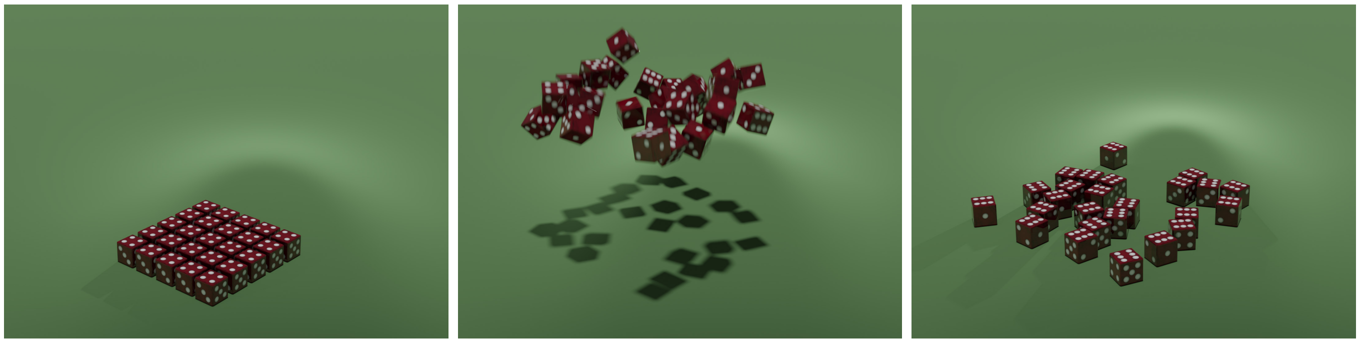
Figure 10: B#ggle: After vigorous shaking, all 25 of these dice land 6 side up (23 transitions).

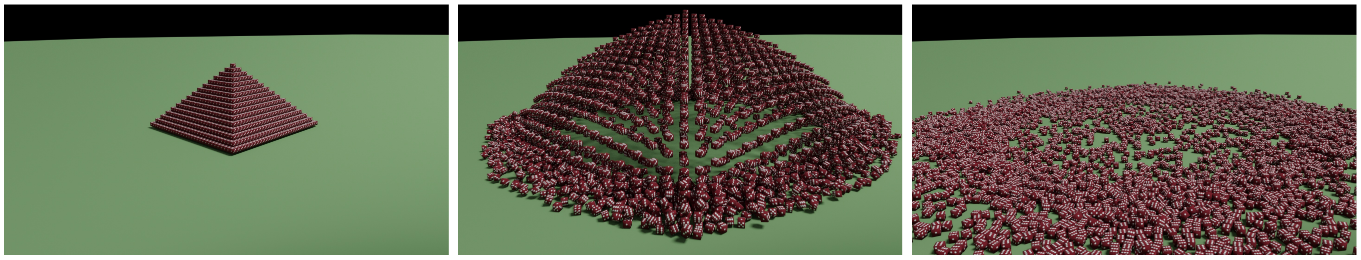

Figure 11: Dice Pyramid: After exploding, the 5456 dice constituting this pyramid all start and stop 6-side up. ("It could happen," they said.)

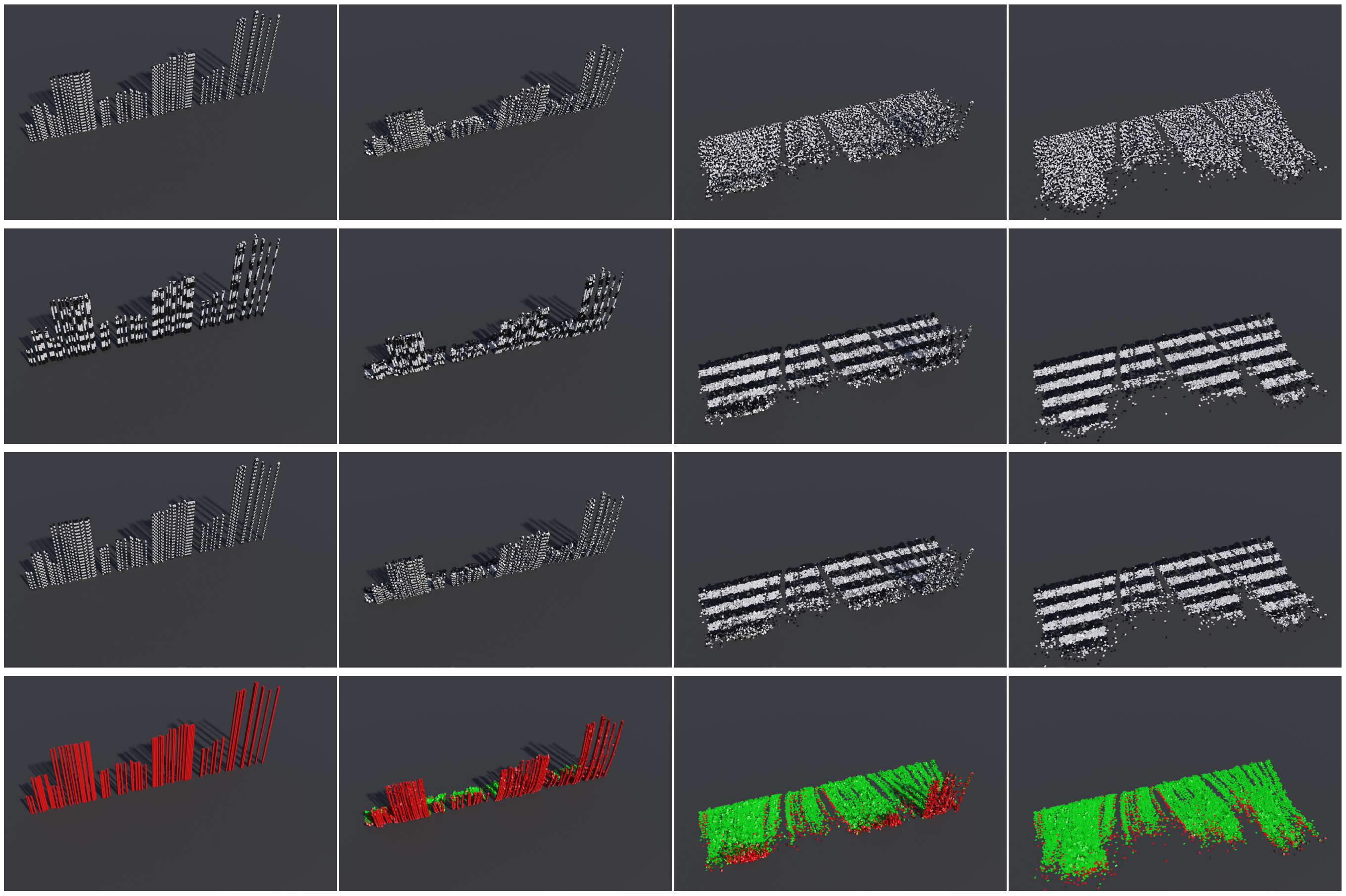

Figure 12: Faulty Card Towers: Card colors are transitioned at the first moment they are not visible from the camera viewpoint. (Top) Initial color constraint. (Middle) Final color constraint. (Bottom) Visibility-shy color transition. (More-bottom-still) Transition visualization changing from red to green. In this conservative run, only cards that were completely hidden were transitioned (Houdini Bullet simulation of 16,190 rigid cards).