Physink: sketching physical behavior

Citation

As Published
http://dx.doi.org/10.1145/2508468.2514930

Publisher
Association for Computing Machinery (ACM)

Version
Author's final manuscript

Accessed
Thu Dec 06 07:23:25 EST 2018

Citable Link
http://hdl.handle.net/1721.1/86117

Terms of Use
Creative Commons Attribution-Noncommercial-Share Alike

Detailed Terms
http://creativecommons.org/licenses/by-nc-sa/4.0/
ABSTRACT
Describing device behavior is a common task that is currently not well supported by general animation or CAD software. We present PhysInk, a system that enables users to demonstrate 2D behavior by sketching and directly manipulating objects on a physics-enabled stage. Unlike previous tools that simply capture the user’s animation, PhysInk captures an understanding of the behavior in a timeline. This enables useful capabilities such as causality-aware editing and finding physically-correct equivalent behavior. We envision PhysInk being used as a physics teacher’s sketchpad or a WYSIWYG tool for game designers.

RELATED WORK
PhysInk’s UI is inspired by systems like K-Sketch [4] and Dragimation [8], which allow the user to record motion by directly manipulating the sketch or drawing trajectories. PhysInk takes these ideas a step further through its ability to understand and enforce physical constraints, establishing the physicality that is tedious to maintain in animation software.

USING PHYSINK
The user starts by sketching objects (rigid bodies) and constraints (joints, ropes, springs). Strokes are recognized by an extension of PaleoSketch as circles, polylines, helixes or X-symbols; these primitives are then interpreted as objects or
constraints, based on their context in the sketch. For example, a circle on its own is a round rigid body, where a small circle inside an existing body is a pin joint. The user can also sketch polygonal objects, anchors, ropes and springs.

The user can then easily demonstrate behavior by manipulating sketched objects, but only within the indicated physical constraints. Concurrent movement of multiple objects can be recorded by rewinding, then demonstrating new movement while previously recorded trajectories are played back. The enforced physical constraints result in an intuitive interface for demonstrating movement and contact, where objects respond to collisions and forces as the user expects. For game designers and others, this opens the door to designing by demonstration, rather than by tweaking structural parameters.

THE TIMELINE
PhysInk records the user’s interactions in a timeline - a causally-linked graph of physical events - which can be rewound, played back and edited. There are four types of events: (1) start: used to represent exogenous forces that initiate the behavior, (2) movement: the trajectory of a single object, (3) contact: marking the collision of two or more objects, and (4) end-contact: marking the ceasing of contact between two or more objects.

In our system events capture both literal and qualitative geometry. For example, a movement event records an object’s demonstrated trajectory as well as its relative motion (left, right, above or below nearby objects). A contact event records the collision location, as well which sides are in contact (e.g. ball against top of box). This representation of events, focusing on causality and qualitative geometry, is intended to capture a higher-level, qualitative understanding of the behavior similar to how a user might think of it.

CAUSALITY-AWARE EDITING
The timeline allows the system to reason about the demonstration at a more abstract level than frame-based animation. Consider a pinball game designer who rewinds a description, editing a ball’s trajectory so that it no longer collides with a bumper. In a traditional animation tool, the bumper would still move, while PhysInk uses its understanding of causality to propagate the change through the timeline, deleting events that should no longer occur (the bumper collision) and preserving those that are unaffected. In this way, PhysInk makes it easy to rewind and explore alternative behaviors.

FINDING EQUIVALENT REALISTIC BEHAVIOR
The timeline is also used to ‘clean-up’ the user’s demonstration, producing an equivalent behavior that is physically correct. This is achieved by searching for physical parameters that lead to a simulated timeline that most closely matches the demonstrated timeline. This behavior will be physically correct, because it is produced purely by the physics engine.

PhysInk does this with an exhaustive search, varying all physical parameters (e.g. initial velocities, friction coefficients), except for sizes, positions and orientations to ensure that the sketch’s static appearance does not change. A simulated timeline is produced for each set of parameters and compared to the demonstrated timeline, using a qualitative and quantitative distance. The qualitative distance measures the topological similarity of the two timelines, where pairs of events are considered equivalent if they share the same type (e.g. movement or contact), objects and qualitative geometry. The quantitative distance measures visual similarity, and is computed by summing Euclidean distances between trajectories of the same object in the two timelines. The simulated timeline with the lowest distance scores provides a physically-correct behavior that most closely matches the user’s demonstration.

USE CASES AND FUTURE WORK
PhysInk’s natural UI for describing physical behavior and the features enabled by its timeline make it a potentially useful tool for design-by-demonstration and education. A mobile game developer could design a character’s movement through the world by demonstrating it, rather than fiddling with physical parameters. A physics teacher could quickly sketch scenarios and demonstrate concepts on the physics-enabled stage, then query physical parameters to answer questions like: what initial velocity will allow the ball to clear the wall?

We would like to extend PhysInk to better support these applications. For example, as a game design tool, there should be an entry point for player interactions in the behavior demonstration process. Also, the user should have control over which parameters are fixed while others are searched. Importantly, we plan to conduct user studies to evaluate PhysInk’s usefulness in these applications.

CONCLUSION
We have presented PhysInk, a system for describing physical behavior by demonstration, where users sketch and directly manipulate objects on a physics-enabled stage. Demonstrations are captured as timelines of events, leading the way to causality-aware editing and finding physical parameters that drive the demonstration, which may be useful in physics classrooms and game design.

REFERENCES