2.5D Simulated Keyframe Animation in Blender

Ilene E
ilene.e.1129@gmail.edu
Princeton University
Princeton, New Jersey, USA

Nora S. Willett
noraw@pixar.com
Pixar Animation Studios
Emeryville, California, USA

Adam Finkelstein
af@cs.princeton.edu
Princeton University
Princeton, New Jersey, USA

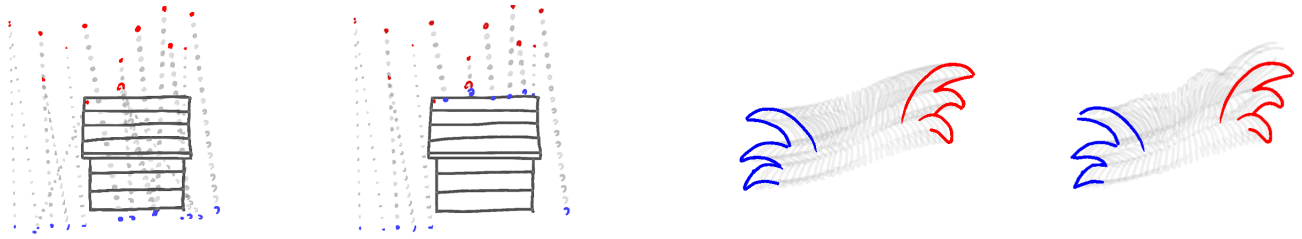

Figure 1: Ghosted frames from the snow collision example (left) and the rabbit fur wind example (right).

ABSTRACT

3D animation requires specialized skills and tends to limit creative expression in favor of physical feasibility, while 2D animation does the opposite. Another duality exists between simulated and keyframe animation. While simulations provide physical believability, keyframes give animators fine timing control. This project seeks to bridge the gap between these approaches to animation: leveraging the expressiveness of 2D animation, the robustness of 3D environment and camera movement, the physical feasibility of simulation, and the control of keyframing. To this end, we present a 2.5D animation interface that takes 2D drawn keyframes and 3D context (object, environment and camera movement) to generate simulated animations that adhere to the user-drawn keyframes.

CCS CONCEPTS

• **Human-centered computing** → **User interface programming**.

KEYWORDS

2D animation, inbetweening, keyframe animation, physical simulation

ACM Reference Format:
Ilene E, Nora S. Willett, and Adam Finkelstein. 2021. 2.5D Simulated Keyframe Animation in Blender. In *The Adjunct Publication of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21 Adjunct), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3474349.3480222

UIST '21 Adjunct, October 10–14, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8655-5/21/10…$15.00
https://doi.org/10.1145/3474349.3480222

1 INTRODUCTION

3D animation offers countless exciting opportunities. However, it requires significant specialized skill and can limit artists' expression in order to adhere to physics principles and object consistency. On the other hand, 2D animation allows full artistic expression including view-dependent inconsistencies and physics-defying motion, but requires arduous frame-by-frame drawing and mastery of principles of motion.

Simulation-based techniques tend to yield realistic movements in animation, but sacrifice users' fine control, especially on timing. Keyframe animation returns this timing control to the animator while producing less physically compelling movements than those grounded in simulation [4].

Our animation interface produces 2.5D simulated keyframe animation. It works by distorting 2D strokes within 3D space given camera movement. Furthermore, it supports collision and wind simulations working with guidance from keyframes, thus allowing animators to exercise timing control while also taking advantage of simulation-based movement enhancement. We present a system that takes input from users through drawn 2D keyframes, (optional) 3D environment objects and (optional) camera control in 3D space. Given the keyframes and camera movement, our system utilizes a novel algorithm to propose stroke correspondences, then collaborates with the user to construct a final stroke matching. For each pair of adjacent keyframes, our system then runs physical simulations to enhance the inbetweened animation. Finally, the system outputs an animation that adheres to physical principles as well as the motion described by the user-drawn input.

2 APPROACH

Above is an overview of how users interact with our system. First, they draw 2D keyframes and create a 3D environment. The system then proposes a matching of the 2D strokes to the user, which the user can examine and revise with additional customization. This portion of the process can happen however many times, until the user is satisfied with the proposed matching. The system

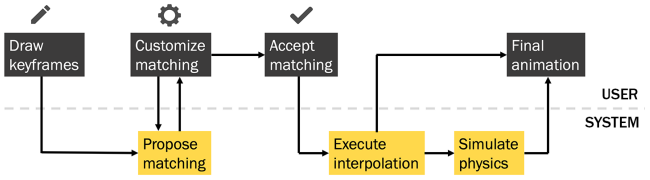

Figure 2: Overview of UI workflow.

executes the linear interpolation and (optional) physical simulation, outputting the final animation inbetweening the original 2D keyframes while colliding with the 3D environment.

3 ALGORITHMS

3.1 Stroke Correspondence

To determine stroke correspondence, we defined an energy value for each potential stroke pairing for strokes A and B:

$$\text{energy}_{A,B} = \text{dist}(\text{centroid}_A, \text{centroid}_B)$$
$$+\text{dist}(\text{shape context}_A, \text{shape context}_B)$$

where shape context was calculated using the method from Belongie et al. [1, 3].

These energy values were used to create a global ranking of all potential stroke pairings. After processing user-specified stroke pairings, we paired strokes based on a greedy algorithm. Using this method, our system can generate emitters and handle input involving: keyframes with different numbers of strokes, more than two keyframes, keyframes with different stroke lengths (paired strokes containing different numbers of points), and keyframes with significantly differing content (e.g., two different poses within a walk cycle or two different angles of a character).

3.2 Simulation

Collision physics were simulated by modeling the scene components using bodies with different motion levels and shape types. Bodies were either static or dynamic in motion: bodies that were unaffected by physics were modeled as static, while those that were meant to react to physics were modeled as dynamic. Furthermore, bodies were modeled with different shape arrangements: circles, polygons, and chains (a connected series of circle bodies).

Wind forces were applied to each individual dynamic body in the scene and altered over time using a modified sine function to mimic wind "gusts". In chains, the wind force magnitudes were also adjusted quadratically to more strongly affect particles farther from the roots.

4 RESULTS

Results are presented in the form of side by side ghosted frames (before physics on the left, physics on the right), with the keyframes shown solidly and color-coded (first keyframe in red, second in blue). To view the full animations, please see our accompanying video.

In the snow example, we modeled the dog house (3D object) as a static polygon shape and the snowflakes as dynamic circle shapes positioned at the calculated stroke centroids (Figure 3). After running collision simulations using Box2D, we outputted the final simulated animation [2].

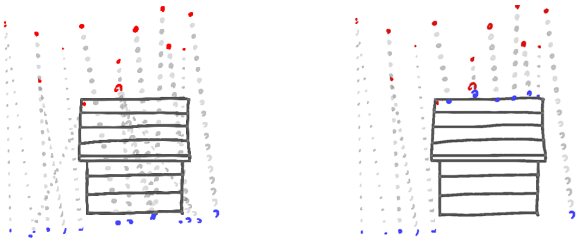

Figure 3: Snow example.

In the rabbit fur example, we simulated an upward wind on the forehead fur using dynamic chains (Figure 4). We offset frames horizontally for additional visual clarity.

Figure 4: Rabbit fur example.

5 CONCLUSION

We present a 2.5D animation interface that automatically generates keyframed and simulated inbetweenings. Potential future work includes improving time performance of the algorithms, expanding the interpolation and simulation repertoire (i.e., providing users the option to ease in/out rather than just interpolate linearly and simulate more forces than just wind and collisions), further refining the stroke correspondence algorithm, and incorporating simulated emitters.

REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha. 2002. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 4 (April 2002), 509–522. https://doi.org/10.1109/34.993558

[2] Erin Catto. [n.d.]. pybox2d 2.1.0 manual. https://github.com/pybox2d/pybox2d/wiki/manual

[3] Andrey Nikishaev. 2019. Shape Context descriptor and fast characters recognition. https://medium.com/machine-learning-world/shape-context-descriptor-and-fast-characters-recognition-c031eac726f9

[4] 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