Warping the space around an animated object

Daniele Federico*
Dr.D Studios

Damien Fagnou†
Moving Picture Company

Tom Reed‡
Moving Picture Company

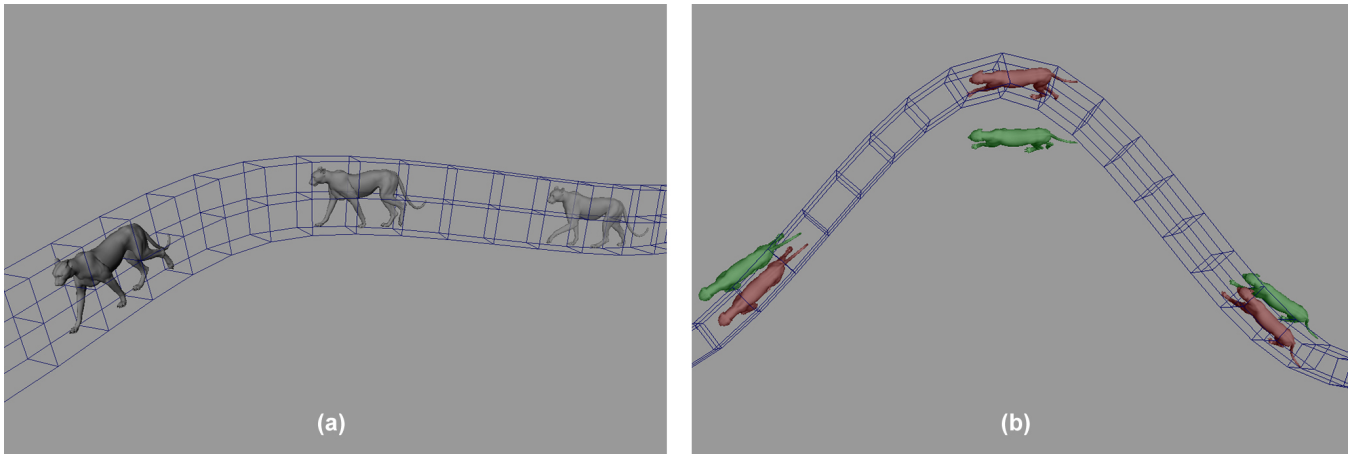

Figure 1: *(a) A warped animation cycle. (b) A cycle deformed with bezier (green) and linear (red) interpolations.*

1 Introduction

The creation of animation clips and the tweaking of existing character animation is often a tedious, time-consuming task, especially in large-scale CG productions.
Traditionally these tasks were achieved by animators manually changing multiple keyframe values for all the relevant animation rig controls. Starting with the idea that a single animated object (e.g. a rig control) essentially defines a time-varying curve in 3D space - where the control points are defined by the keyframe values of the translation channels - we introduce a modeling approach for deforming these conceptual 3D curves. We will talk about our current implementation based on FFDs (Free Form Deformations) as described in [Sederberg and Parry 1986], but we strongly believe the same approach can have many other usages (e.g. modeling tools, collisions and obstacle avoidance).

2 The space warping approach

Motion warping was demonstrated in [Witkin and Popović 1995], where the authors describe an approach for warping the motion of an animated character starting with a pre-existing cycle and providing new animator-defined key poses as inputs. Their algorithm then modifies rotation and position values (when needed) for the skeleton joints in order to pass through the keyframe poses.

The approach we propose is based on applying operators (e.g. deformers) directly on the 3D space in which our animated objects are moving. This leads to a variation of the paths and orientations of the objects over time. In order to find new translation values for each object we map positions from the ”untouched” 3D space to the deformed one. In the same way, the axes of the transformation matrices are deformed and then orthogonalized before performing any conversion to Euler angles or quaternions. In our system we have so far chosen not to consider any scaling factors.

At MPC as a proof of concept, our initial implementation entailed applying FFDs on existing animation clips in order to create new motions. These resulting motions were then used both in ALICE (MPC's crowd engine) and directly in the animators' Maya sessions. The tool can be used on one or more characters at the same time with a minimal impact on the scene's performance. The lattice for the evaluated character is generated by the tool itself in order to completely encompass the controls for the entire input animation. If an object is evaluated outside of the lattices boundaries unpredictable results may occur.
After the first tests we found that for modifying the animation for a character, it is enough to apply FFDs only to the controls which move in world space, such as IK or global controls. However our tool also works well for finding new local values for parented transforms. In fact our tool always generates a new orthogonalized world matrix, which in this case must be multiplied by the inverse of the parent matrix; therefore, this approach can be also used with FK controls.
Our algorithm works with both bezier and linear interpolations. The former is used for a smoother result, while the latter is employed whenever the user needs more control over the character.
An undesired effect may occur whenever the lattice cells get too stretched, in fact this can lead to the typical ”foot-skate” effect. Our tool cannot fix this situation; what we have done instead is to provide users with some visual warnings to indicate when the cells shrink or become too large. In this way undesired ”skating feet” can be prevented.
Once the animator is happy with the achieved result, a baking process is run. The baking process attempts to change only the values of existing key frames to align the original animation with the deformed one, but will fall back to baking entire channels when necessary. We provide other utilities for simplifying the baked animation curves to our animators, so they won't have the heavy task of cleaning the result.
This approach has been proved to be very powerful for two reasons:

1. its evaluation is never dependent on the next or previous configurations for the considered object. This leads to a solution which depends only on the current input matrix and the shape of the lattice.
2. it provides local control over motion. Each CV of the lattice can be animated and affect only certain zones of our characters at a certain time.

We have used this tool in production on both primary and secondary characters achieving very satisfactory results. Specifically, our implementation has been proved to be very suitable for modifying basic animation cycles into blocking and rough animations for hero characters. Whereas animators would previously spend a day or more generating new cycles, we have observed them achieving the same result in only two hours using this tool.

Thus far, we have found this solution to be efficient and fairly computationally inexpensive. Looking forward, we believe this approach could achieve its best usability and performance in conjunction with a good animation layer system.

References

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4, 151–160.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. *Computer Graphics 29*, Annual Conference Series, 105–108.

*e-mail: df@danielefederico.it
†e-mail:damien-f@moving-picture.com
‡e-mail:tom-r@moving-picture.com

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007