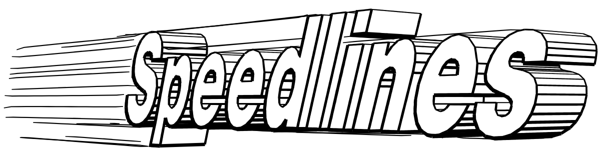


Depicting Motion in Motionless Pictures

Maic Masuch, Stefan Schlechtweg and Ronny Schulz

How do we present the motion of objects in computer generated still images? Generally, we don't. Or, if we do, we use motion blurring to simulate a real-world camera [PC83]. Besides, it costs a great deal of extra rendering time, and the only thing it does is blur the objects so that their contours are unrecognizable. The goal to convey information about the movement *per se* is not entirely met. If, however, we consider the application of non-photorealistic rendering techniques, it is promising to adopt successful illustrative techniques from comics to depict past and future motions of objects in a single image.

Presenting Motion

Speedlines are an important stylistic element in comics, and although they are not based on an exact physical model, their use is well known to all of us [McC93]. Starting from the very first cartoons *Little Nemo* and *Krazy Kat*, speedlines have always depicted past motion.

Another technique to illustrate the movement of an object is to indicate former positions by drawing repetitions of its contour. The contour consists of the outline and inner lines. Often, only part of the outline is drawn in order to maintain clarity and simplicity of the depicted scene.

If the speedlines are mirrored and only one of them is drawn with an arrow, it depicts a possible future motion, as commonly used in technical documentations. Although these techniques were developed primarily for still images, their use has also been adapted in many classical animations [TJ81].

Generating Speedlines, Contours and Arrows

Our approach to generate speedlines, arrows, and repeated contours uses a polygonal 3D model along with keyframe data of an animation as input. The process of generating illustrative speed elements is implemented as post-processing, since it is important to know the final appearance and the z-ordering of the objects to depict. After stepping through a rendering pipeline for line drawings (cf. [Str98]), the system uses 3D information gathered during the rendering in order to establish the appropriate visibility and the perspective correctness of the speedlines. Here, a 2D-based approach would be futile, since we have to keep track of the contour of every single object. Then the motion of the object is scanned using given keyframe information, and a motion path is calculated. After that, we know all scene elements with their corresponding 3D and motion information to draw illustrative speed elements.

As these elements were invented by human artists, their generation requires some heuristics about where and how to draw them. In general, speedlines and contour lines

- are drawn in the opposite direction of the movement (thus reaching into the past),
- start at "characteristic" points of the moving object,
- embrace the minimum and maximum extent of the object,
- are more or less equally sized, shaped and directed, without intersecting each other,
- are uniformly, but not too regularly distributed.

Candidates for starting points of speedlines are vertices of the 3D mesh which – after their projection in 2D – yield the outline of the object. The algorithm divides the shape into a number of stripes, whereas each stripe can hold one speedline. If a stripe contains no such vertex, additional vertices are generated on the object's outline. If we do not restrict the algorithm to the outline and also take into account inner contour lines, the result can be further improved. In addition, the speedlines should not stick to an object, i.e., the algorithm draws them with an offset. Figure 1 illustrates different categories of illustrative speed elements: speedlines, full and partial contour repetitions, and arrows.

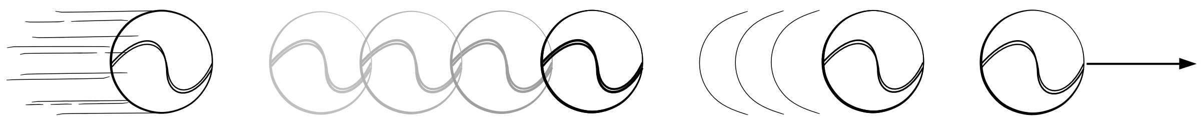


Figure 1: A thrown ball drawn with randomized, segmented speedlines, repeated, fading contours, partial contour repetitions, and a motion arrow.

As we still hold all necessary scene data, the computation of these effects takes only a split second, even for larger scenes. Thus it is quite easy for a user to apply different techniques interactively to an object and experiment with a combination of speed elements as shown in Figure 2.

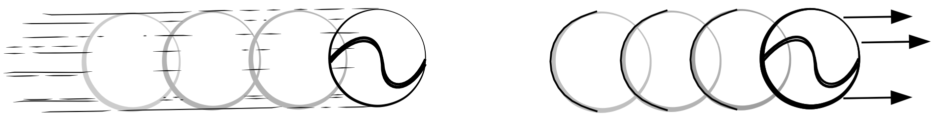


Figure 2: A combination of speed elements can enhance the impression of motion.

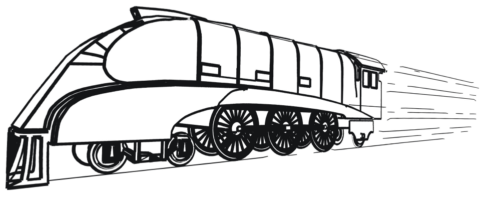
Drawing Lines with Style

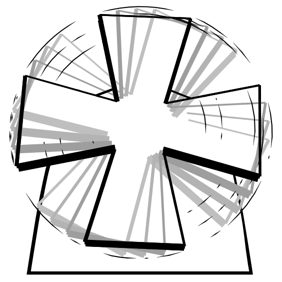
The output of the renderer consists of a vector-oriented description of the image, namely all visible lines. These lines are drawn using line styles which, in turn, simulate hand-drawn pen strokes by the superposition of the drawing path and the chosen style deviations. The drawing path itself consists of interconnected visible lines, whereas the style is a parametric description for characteristic line deviations.

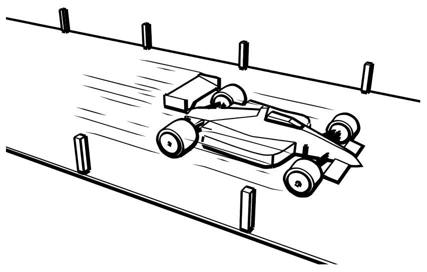
Although our system supports parameter control in every detail, only very little user interaction is necessary, since nearly all speedline parameters can be computed based on the animation data. Therefore, length, width, distribution, number, etc. are determined by applying given default rules.

Some Results

The pictures below show different models with speedlines applied to them. We have chosen to render quite simple models to concentrate on the effects of speedlines and contour repetitions. Also, we applied line styles only to the speed elements.







Speedlines, in general, are not very well suited for movements directed towards the viewer. Horizontal movements like those of the train evoke speedlines which are not perpendicular to the viewing plane and thus are better to visualize. Motion blurring the wings of the windmill would smear their shape, whereas applying speedlines and contour repetition renders them still recognizable. Further examples including a short animation can be found at `http://isgwww.cs.uni-magdeburg.de/~masuch/gallery.html`.

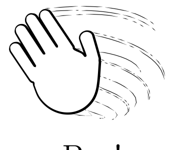
References

[McC93] Scott McCloud. *Understanding Comics - The Invisible Art.* HarperCollins Publishers, Inc. New York, 1993.

[PC83] M. Potmesil and I. Chakravarty. Modeling Motion Blur in Computer-Generated Images. In *Computer Graphics (SIGGRAPH '83 Proceedings)*, volume 17. ACM SIGGRAPH, ACM Press, 1983.

[Str98] Thomas Strothotte. *Computational Visualization. Graphics, Abstraction and Interactivity.* Springer Verlag, Berlin, Heidelberg, New York, 1998.

[TJ81] Frank Thomas and Olli Johnston. *The Illusion of Life: Disney Animation.* Hyperion, 1981.



Bye!