University of Miskolc

Faculty of Mechanical Engineering and Informatics
Institute of Automation and Infocommunication

TDK thesis

Emergent Animation Systems: from interpolation to procedural motion

Author:
Konrád Soma Kiss
BSc Student in Computer Engineering

Supervisor:
Dr. Attila Károly Varga
Associate Professor

Miskolc, 2025

Contents

1 Introduction

When the term “animation” or “motion graphics” is mentioned, people often think of the process of creating animated films or cartoons. This usually refers to hand-drawn animations or, more recently, computer-generated imagery (CGI) and visual effects. However, the terms "animation" and "motion graphics" have much broader meanings.

As Adobe, an industry leader in digital media, defines it: "Motion graphics are essentially 'graphics with movement'." [1] This definition aligns more closely with the theme of this thesis. What are procedural animations? What systems can be used to create them? What are rule-based animation systems, and what emergent behaviors can be achieved that are unforeseen by the designer?

The premise is that everything that moves on the screen is animation, and everything that is animated can be considered motion graphics. This thesis will delve into the different algorithms, aspects, use cases, and applications of these systems, and how they are used throughout the digital world, even though we might not notice them at first glance.

The basic hypothesis is that we can create complex movements, behaviors, and animations by defining simple rules and systems. In other words, the less artistic control we have over the final result, the more we can rely on the system to generate interesting, natural-looking motion.

2 Background

To understand the concepts discussed in this thesis, it is essential to have a basic understanding of vectors, matrices, affine transformations, and some principles of computer graphics. An excellent resource for this can be found in the book "Learn OpenGL - Graphics Programming" by Joey de Vries [2]. This book provides a comprehensive introduction to the mathematical foundations of computer graphics, which are crucial for understanding procedural animations and motion graphics.

My recommendation is to read sections 8 through 10 of the book, which covers all the necessary mathematical elements required to proceed.

In addition, it is beneficial to have a basic understanding of programming, especially in JavaScript, as this thesis will include code examples and implementations in this language. I will try to explain everything in mathematical terms first, and then provide the code examples to illustrate the concepts. This approach will help bridge the gap between theory and practice, making it easier to grasp the underlying principles of procedural animations and motion graphics.

3 Methodology

There are some major measurement methods that I will use to evaluate the results of different techniques and algorithms. These methods will help in understanding the performance, repetition, and overall "feel" of the animations created by the systems. While some of these measurements are subjective, they will provide a good basis for comparison.

3.1 Deviation

Deviation ($\bar{D}$), as defined by me in these measurements, is a measure of how much the actual motion deviates from the expected or desired motion. It can be quantified by calculating the difference between the actual position of an animated object and its expected position at a given time.

I will define 1 as the maximum deviation, where the motion has no correlation with the expected motion, and 0 as the minimum deviation, where the motion is the exact same as the expected motion.

I will not provide an exact formula here, as it will depend on the specific use case.

The expected position will usually be a linear interpolation between two keyframes, while the actual position will be determined by the algorithm used to generate the motion. This means that easing functions, procedural noise and any other factors that affect the motion will be taken into account.

This statistic may be a bit biased towards stiff, robotic motions, as they will have a lower deviation from the expected motion. However, it is still a useful metric to evaluate the overall smoothness and naturalness of the motion. There will be a lot of techniques where this will be omitted entirely, because the motion is purely procedural and we cannot define an expected motion. In these cases, we will state that the deviation metric is 1.

3.2 Repetition

Repetition (R) is a measurement of how often of how prone an algorithm is to repetition. This will be a value between 0 and 1, where 0 means the algorithm is purely chaotic, while 1 means the algorithm is completely deterministic and will always produce the same motion for the same input. Of course, due to the pseudorandomness of computers, we could achieve deterministic results for every algorithm; therefore, as an extra rule, I will always use a different seed for each run of the algorithm.

3.3 Control

Control (C) is a measure of how much control the designer has over the final result of the animation. This is a subjective measurement, with these possible values: "none", "low", "medium", "high" and "total". This represents how much influence the designer has over the final result of the animation. A stiff motion from point A to point B would have a "total" control, while a physics simulation would have "low" control, as the designer can only influence the initial conditions and the parameters of the simulation, but cannot control the final result.

3.4 Feel

Feel (F) is another subjective measurement, which will be based on my personal experience and the feedback of others. It will be a value between 0 and 1, where 0 means the motion feels unnatural and robotic, while 1 means the motion feels natural and fluid. This will be based on my personal experience with the animation, as well as the feedback of others who have seen the animation. The pool will not be large, but I will try to get feedback from as many people as

possible. This will be a good indicator of how well the algorithm works in practice, and how well it can create natural-looking motion.

3.5 Performance

Lastly, a purely objective measurement: performance ($P(...)$). This will be measured in ms, and will be the time the algorithm takes to calculate the position and rotation of the objects for the next frame. Every algorithm mentioned here will be tuned for real-time performance, so the goal is to achieve a frame rate of at least 60 FPS, which means the algorithm should take no more than 16.67 ms per frame (with leaving some room for rendering).

If other parameters, like the number of objects and such heavily influence this parameter, I will define an extrapolated function from many measurements. It is similar to the "Big O" notation. It will always be a function defined by me, that tries to estimate the time it takes to run the algorithm for a given input.

3.6 Summary

Measurement	**Range**	**Example**	**Description**
Deviation	0 to 1	$\bar{D} = 0.2$	How much the actual motion deviates from the expected motion
Repetition	0 to 1	$R = 0.9$	How prone the algorithm is to repetition
Control	...	$C =$"high"	How much control the designer has over the final result
Feel	0 to 1	$F = 0.2$	How natural the motion feels
Performance	$P(...)$	$P(n) = n \cdot k$	The time the algorithm takes to calculate the next frame (where n is the number of objects and k is a constant cost)

Table 1: Summary of measurement methods

4 Implementation

Some text here.

5 Algorithms

Some text.

6 Results

Here will be some results.

7 Conclusion

Here is a conclusion.

References

[1] Adobe. *What Is Motion Graphics?* Accessed 08. 05. 2025. URL: https://www.adobe.com/uk/creativecloud/animation/discover/motion-graphics.html.

[2] Joey de Vries. *Learn OpenGL - Graphics Programming*. Kendall & Welling, 2020. URL: https://learnopengl.com/book/book_pdf.pdf.