A Conceptual Framework for Procedural Animation (CFPA)

Dong Joo Byun
Walt Disney Animation Studios

SIGGRAPH '18 Courses, August 12-16, 2018, Vancouver, BC, Canada
ACM 978-1-4503-5809-5/18/08.
10.1145/3214834.321483535

Abstract

This course presents a conceptual framework for procedural animation (CFPA) that defines and describes common language for a fundamental timing definition that can be used to design and drive procedural animation. The course will use both test and real production cases to illustrate these concepts. By following the CFPA, users can set up procedural animation rigs and tools in a highly organized and modularized way to facilitate authoring and reuse.

About the Lecturer

Dong Joo Byun

Walt Disney Animation Studios
500 S. Buena Vista Street. Burbank, CA 91521
+1-818-967-8749
dong.joo.byun@disneyanimation.com

Dong Joo (DJ) Byun is an effects animator at Walt Disney Animation Studios. Byun has contributed his talents to 5 animated feature films and 4 animated shorts since joining the studio in 2010. Aside from his production work, Byun's main R&D areas are procedural rule based effects animation tool development and automation / pipeline development. Byun received a B.E. degree in chemical engineering from the Sogang University in 1997, a B.A. degree in architecture from the Hongik University in 2002, and an M.F.A. degree in animation and visual effects from the Academy of Art University in 2010.

Course Overview

Conceptual framework overview (10 min)
An overview of procedural animation. Introduction of conceptual framework of procedural animation (CFPA). The concept of *actF*, *endF*, *lifeF*, *ageF*, and *ageFNorm* will be explained.

Single procedural animation (10 min)
Demonstration of set-up mechanism and usage of CFPA with a roll deformer example.

Procedural animation with multiple inputs (10 min)
Varied procedural animation timing control per geometry with CFPA. Procedural wave deformer example will be presented.

Single input multiple animation (10 min)
Controlling multiple animation with single *ageFNorm*. A flower blooming example will be demonstrated.

Usage of *ageF* (10 min)
Usage of *ageF* for controlling procedural animation, which is activated at *actF* and keeps performing without *endF*. Microbot example will be demonstrated.

CFPA as a substitute for simulation (10 min)
Procedural dynamics and CFPA.

Storage of animation data with static geometry (10 min)
Benefits of storing CFPA data with static geometry.

Hierarchical procedural animation (10 min)
Inheritance of CFPA data through hierarchical geometry.

Non-CFPA-based procedural animation (5 min)
Non-CFPA-based procedural animation and CFPA-based procedural animation.

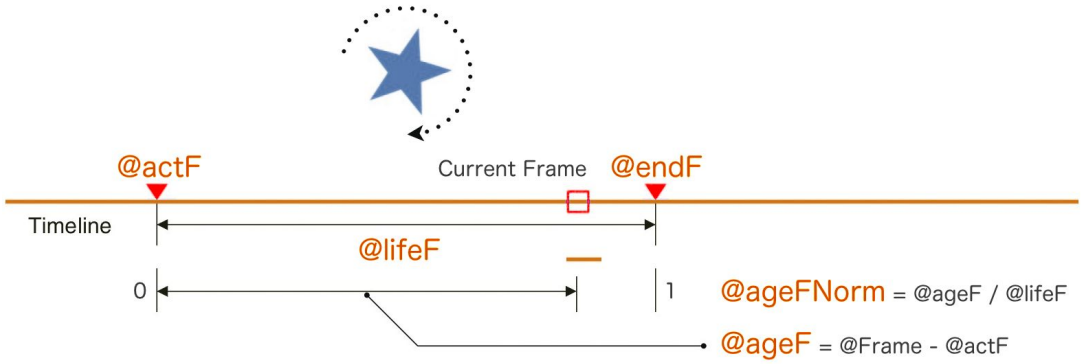
Q & A (5 min)

Introduction

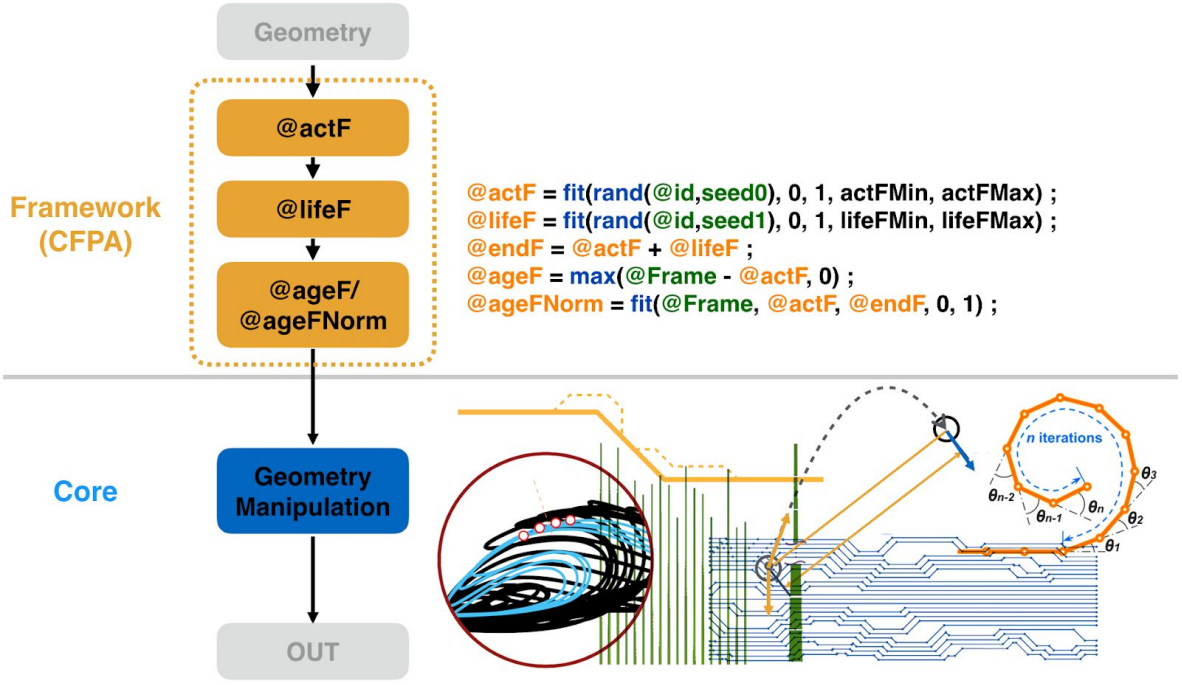
Simulation is the predominant approach used for effects animation to create organic and complex motion, but it has two crucial limitations: it is (1) not interactive, and (2) difficult to stylize. On the other hand, since procedural animation approaches are independent of sequential time steps, they can offer very interactive and intuitive workflows. Procedural animation rules do not always have to be physically accurate: they can be analytic and optimized for specific stylization. Unlike simulation approaches, guidelines and standards have not been developed for procedural animation; as a result, it has been difficult to package up and share procedural animation set-ups. Procedural animation development can be separated into two different parts: geometry manipulation rules and a timing definition. The timing definition is a general guideline for procedural animation development. For example, the wave deformer developed for Moana is a specific geometry manipulation rule, whereas the input timing attributes used to drive the wave deformer are general concepts that are applicable to other scenarios. This course introduces five CFPA based attributes as timing definition components of procedural animation set-ups. For the purposes of this course, We will not cover the details of specific geometry manipulation rules; instead, various manipulation rules will be used as driving examples of how to apply the CFPA in various scenarios.

Conceptual Framework Overview

The CFPA describes a general timing definition which is made of 5 different time-related attributes - *actF*, *endF*, *lifeF*, *ageF*, and *ageFNorm*, where the letter "*F*" in each attribute stands for "frame". These attributes are derived from the activation and end timings of the procedural rules' execution. *actF* is the activation frame, and endF is the end frame. *lifeF* is the lifespan of the animation, and ageF is the age, which is the frame range from actF to the current frame. *ageFNorm* is the normalized age of the procedural rule. ageF is typically used when actF is defined without an *endF*, whereas *ageFNorm* is typically used when both attributes have been defined. This diagram shows how these 5 CFPA attributes are working on the timeline.



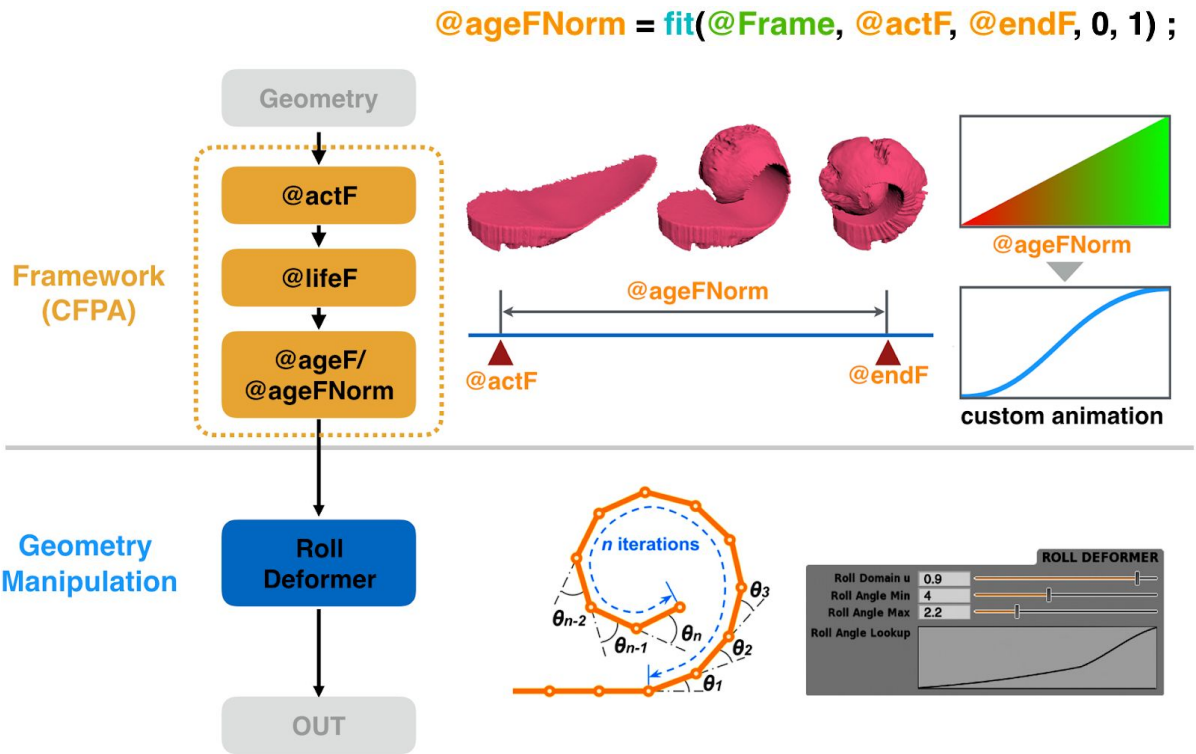
The diagram below conceptualizes a modularized structure of a procedural animation set-up, which is organized with CFPA as a timing definition building block.



Single Procedural Animation

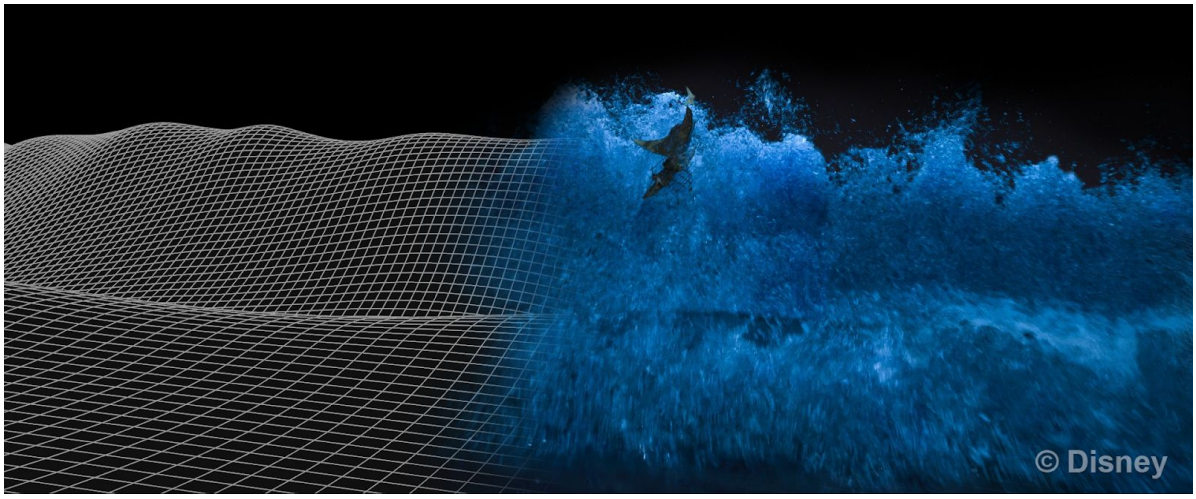
Once *ageFNorm* is defined from *actF* to *endF*, users can apply an animation curve to design the motion. The roll deformer which was developed for an ice cream scooping effects is an example. This example will illustrate how to set up CFPA from the scratch.

In this example, *ageFNorm* is remapped with a lookup curve to apply art-directed timing.

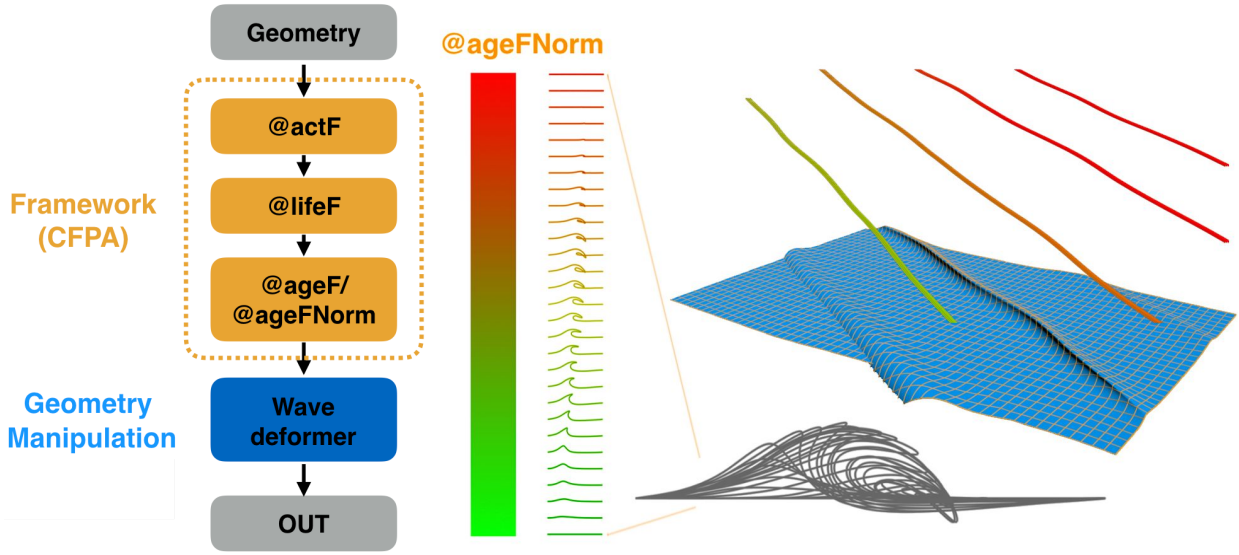


Procedural Animation with Multiple Inputs

Multiple CFPA inputs can drive multiple procedural animations with different timings. In the wave deformer example, CFPA attributes stored in multiple inputs can be used to control the speed, density, timing and composition of procedurally animated patterns.

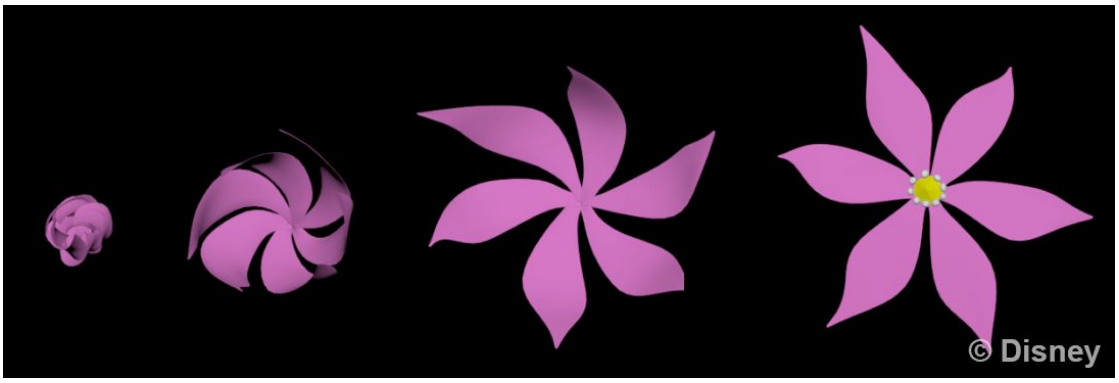


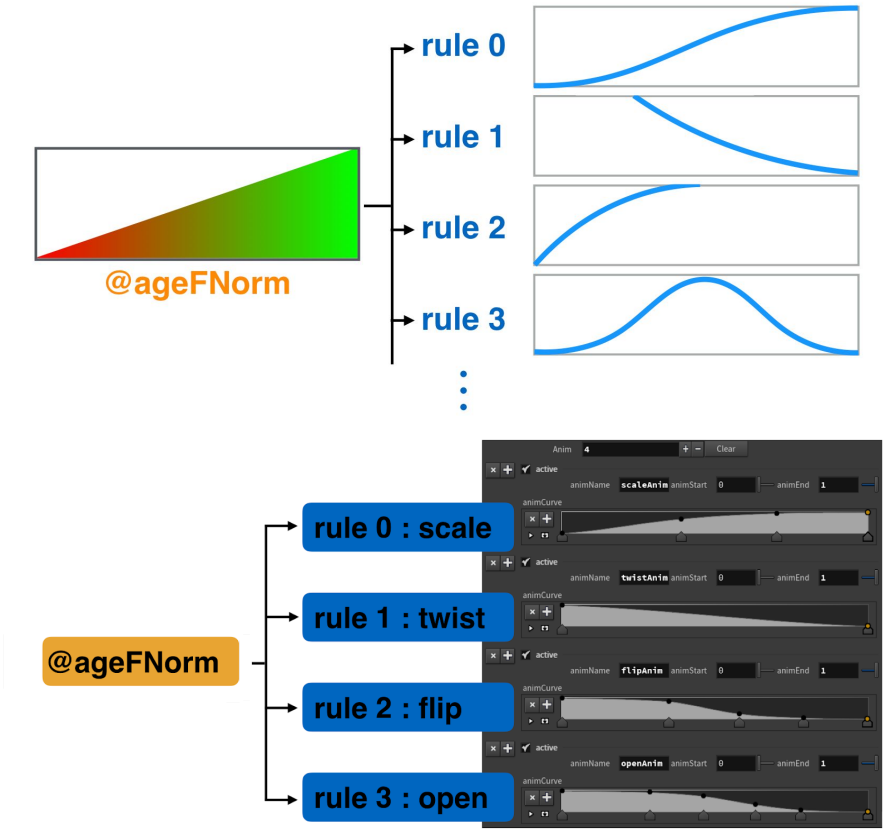
In this example, CFPA attributes are stored in moving curves which represent individual waves. Water surface geometry reads the timing information from these curves, and applies corresponding wave pattern which is provided by a wave deformer.



Single Input Multiple Animation

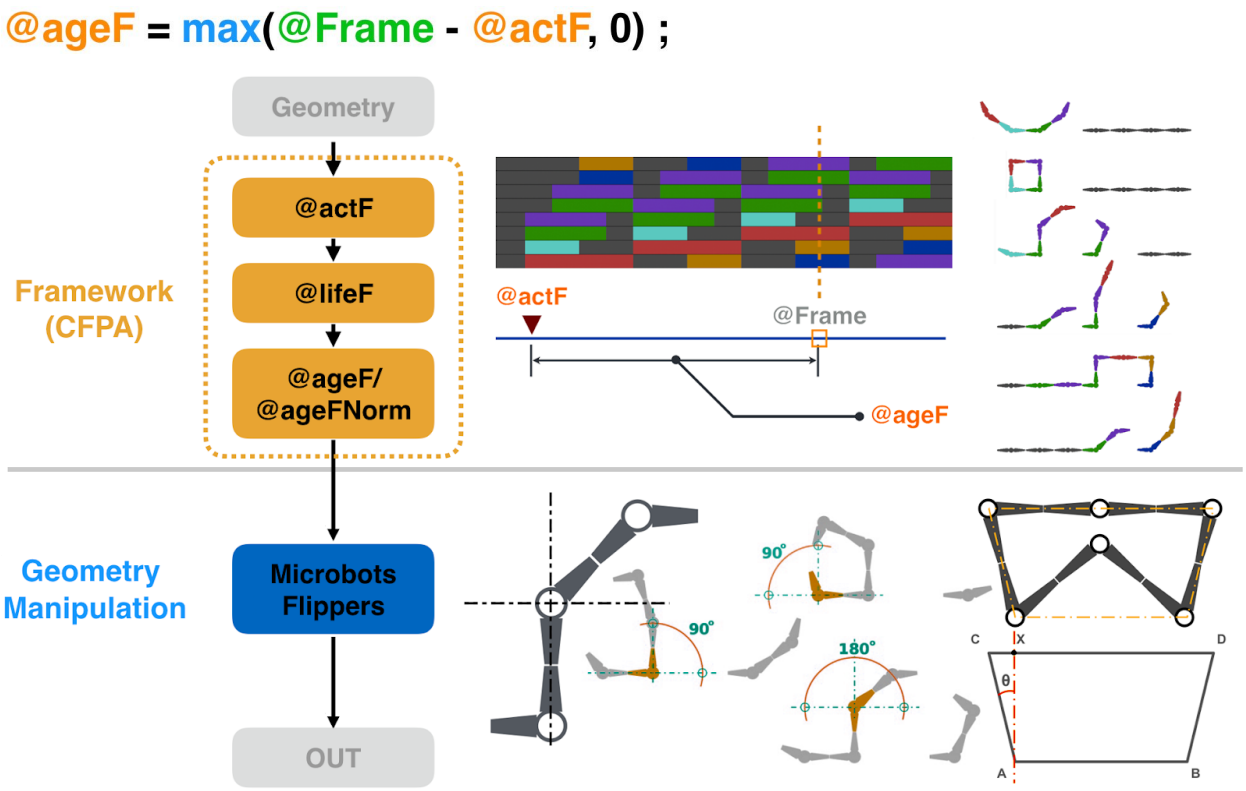
Certain types of effects are defined with the combination of multiple procedural animations. This example demonstrates how a single input, *ageFNorm*, drives the scaling, twisting, flipping and opening of the petals, to generate coherent natural and magical flower blooming effects.





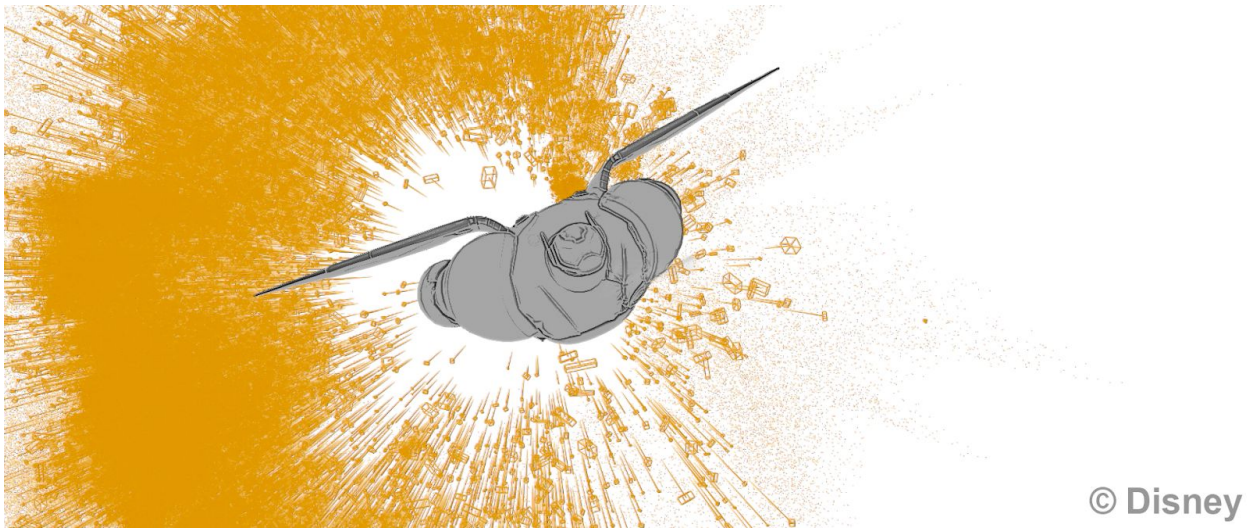
Usage of *ageF*

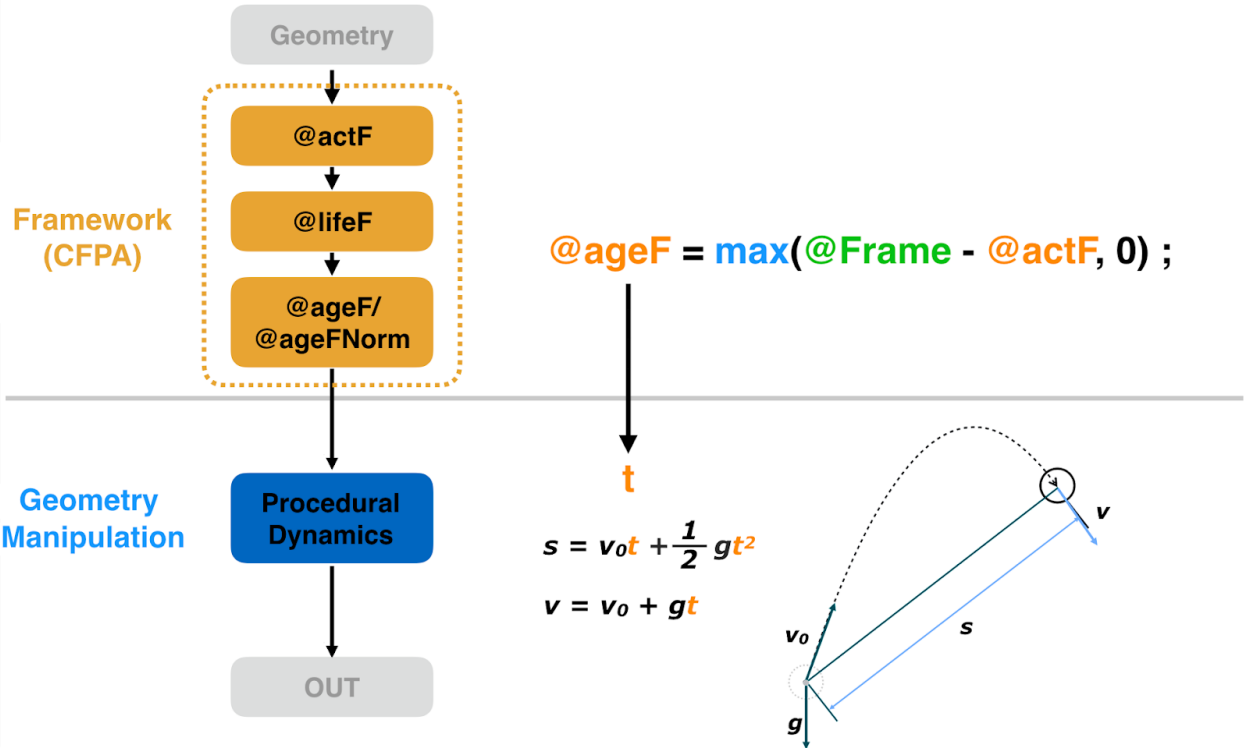
Continuous procedural animations that start at *actF* and have no defined *endF* can be controlled with *ageF*. In the microbots flippers example, microbots are activated at different *actF* values and continue performing flipping motion.



CFPA as a substitute for simulation

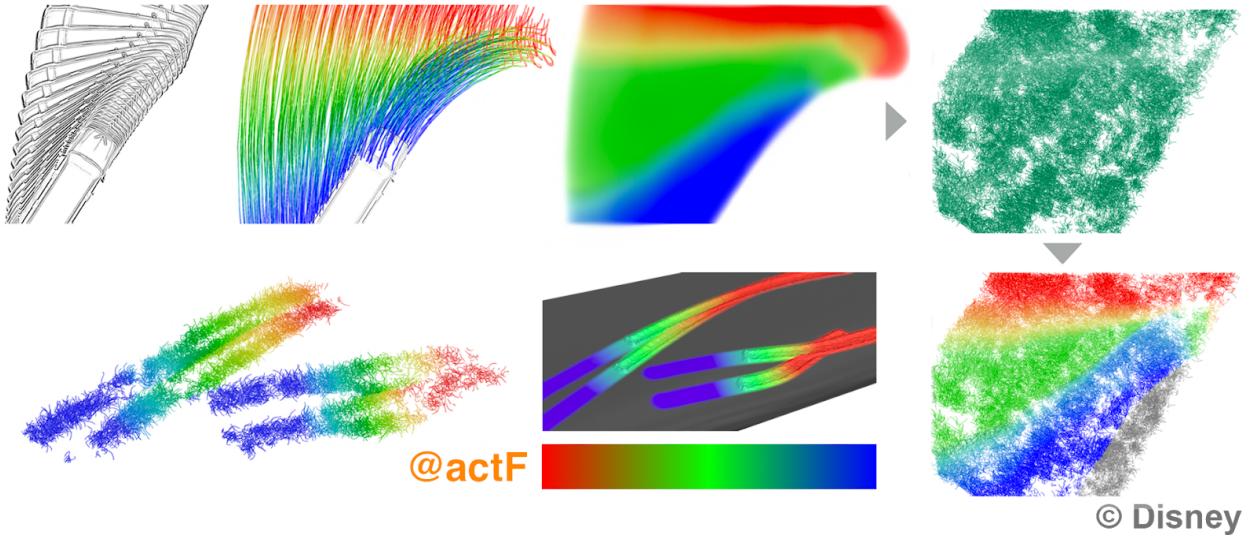
Procedural dynamics show that the proper usage of procedural animation can be a highly optimized substitute for corresponding simulation solutions. Procedural dynamics examples show how kinematic equations from dynamic solvers can be used more intuitively in procedural dynamics by leveraging the CFPA timing definition.

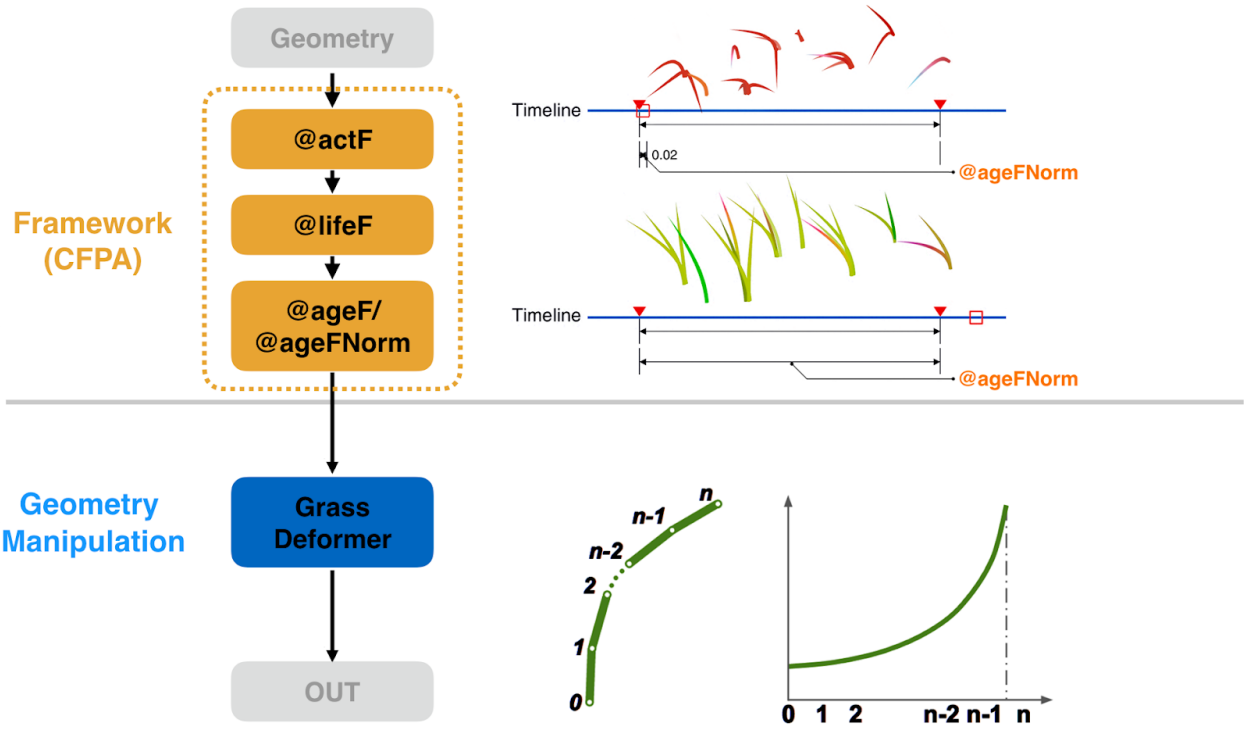




Storage of animation data with static geometry

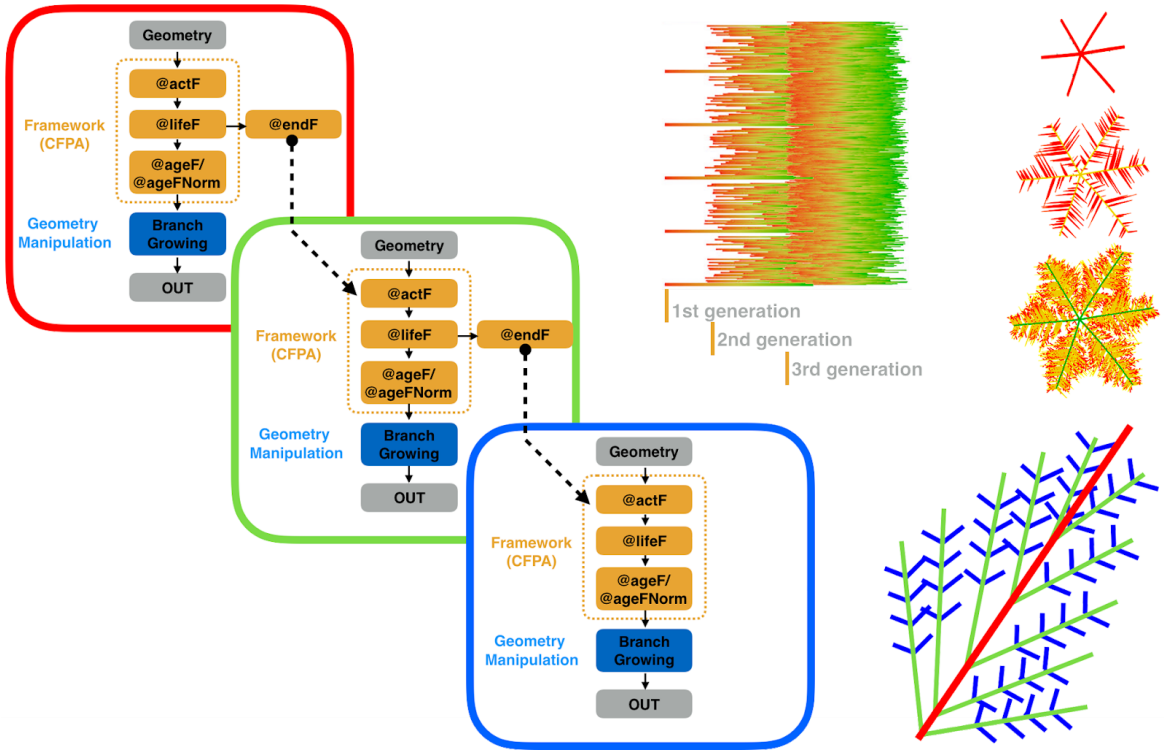
By storing animation data with the static geometry such as volume fields or curves, the geometry cache reading process from disk, can be skipped, which dramatically increases performance. It allows us to drive the procedural animation very interactively and intuitively. Another benefit of this solution is, that since the CFPA data is interpolated when stored in static geometry, we can deal with continuous time. A grass interaction example will serve to demonstrate how we can use this method with the timing definition from CFPA.




Geometry
@actF
@lifeF
@ageF/
@ageFNorm
Framework
(CFPA)
Timeline
0.02
@ageFNorm
Timeline
@ageFNorm
Geometry
Manipulation
Grass
Deformer
OUT
n
n-1
n-2
2
1
0
0 1 2
n-2 n-1 n

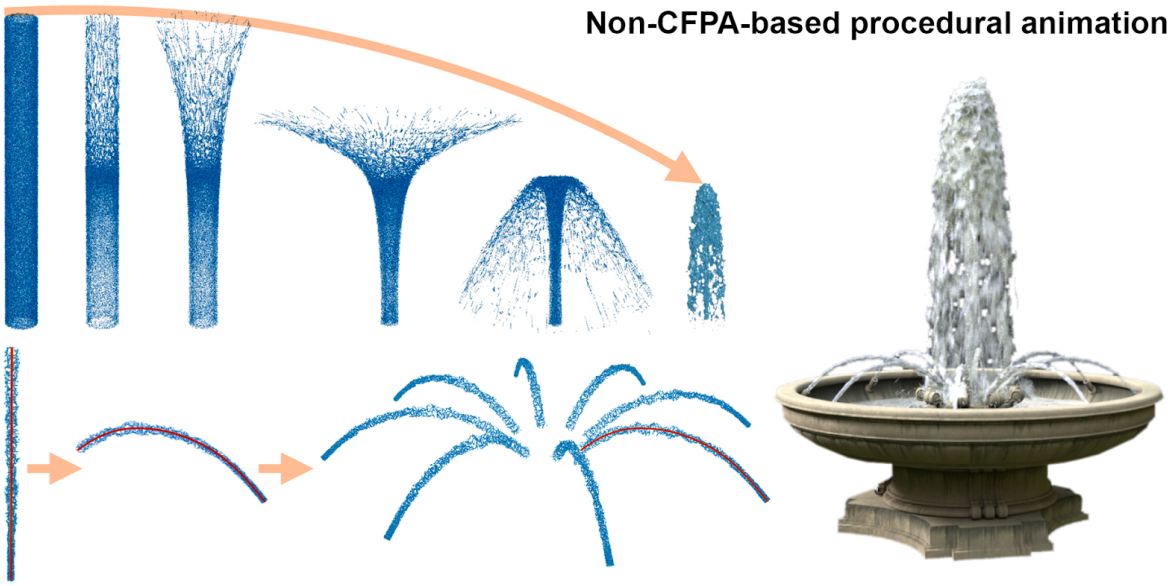
Hierarchical procedural animation

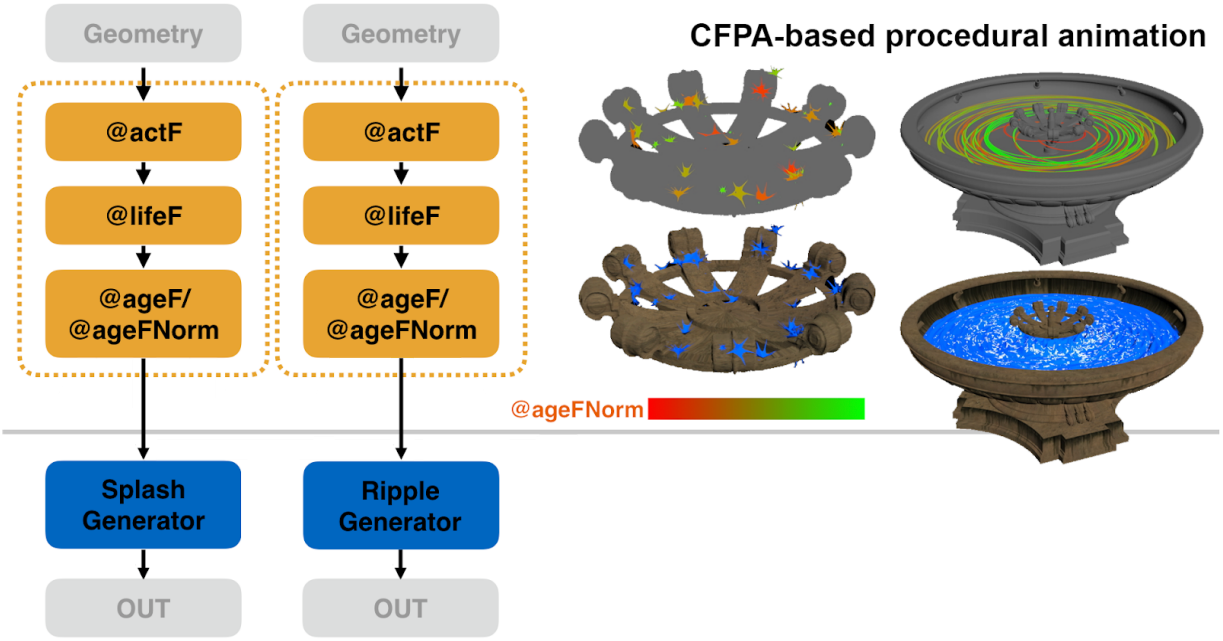
When the geometry is organized hierarchically, the timing definition will be inherited from ancestors to descendants. A procedural snowflake generation example will be shown to explain how this category of procedural animations can be controlled with the timing attributes from the CFPA. This example shows how *actF* can be transferred recursively and sequentially from the 1*st* generation branches into 2*nd* and 3*rd* generations.



Non-CFPA-based procedural animation

Procedural animations that do not have specific activation or termination, but have continuous or repeated patterns of motion, do not require the CFPA. A fully procedural water fountain will be shown as an example. The main and side stream of a fully procedural water fountain will be shown as an example of an animation generated without the CFPA. Procedural water splashes and procedural ripples are other elements and they are generated with the CFPA.





Conclusions

This course is designed to provide the audience with an organized and shareable framework for procedural animation projects, and encourage attendees to research and explore various procedural animation approaches. I hope that this course will stimulate active discussions on how we might apply more stylization and art direction to our work utilizing the CFPA-based procedural animation approaches.