CMPM 120

Animation & Tweens

Objectives

By the end of today you should be able to...

1. Articulate what "tweens" are in animation
2. Apply tweens to your Phaser objects
3. Understand how the principles of animation apply to videogames

Today's code exercise

https://github.com/ikarth/animation

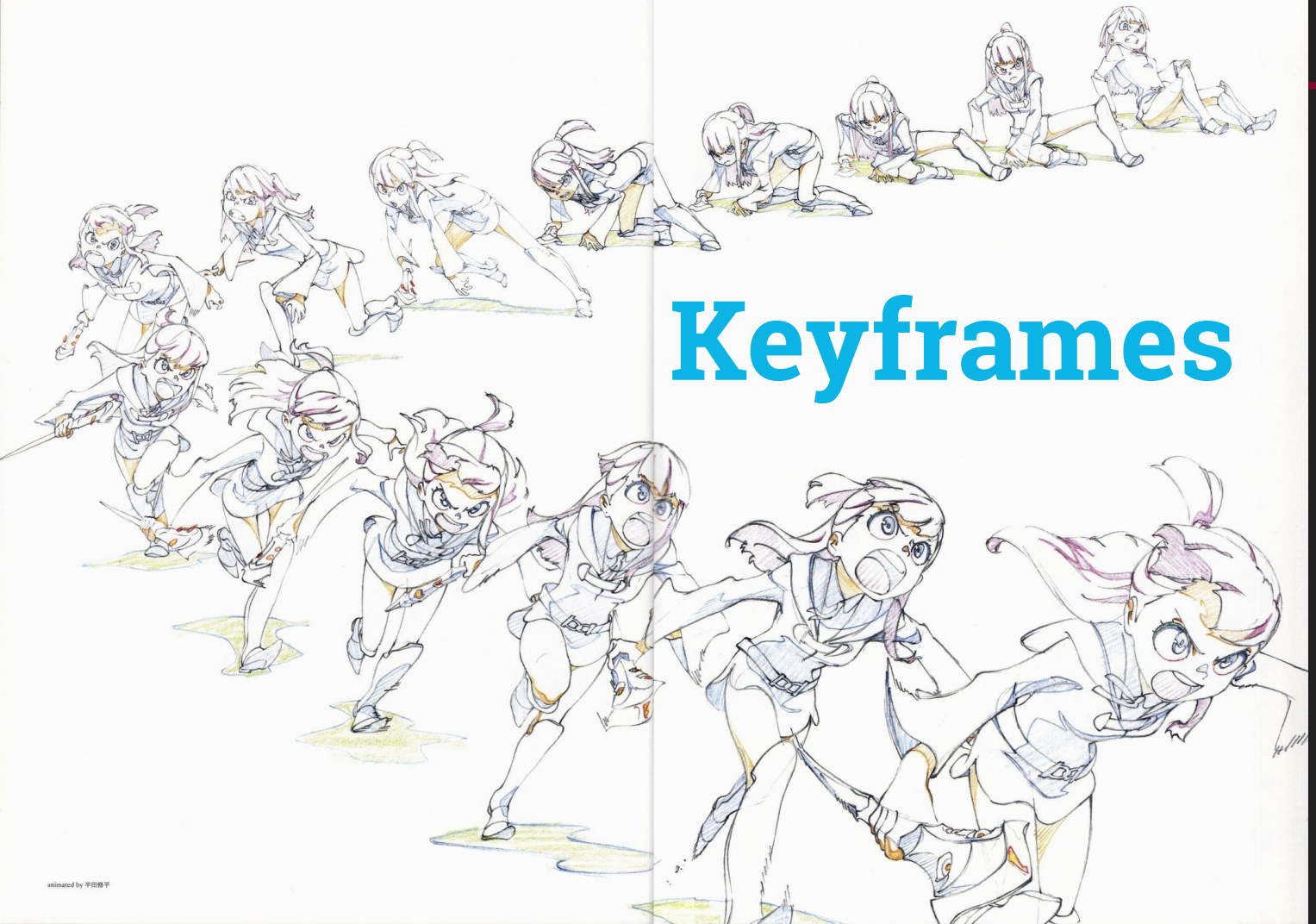
Akira (1988) animation cel

Frame

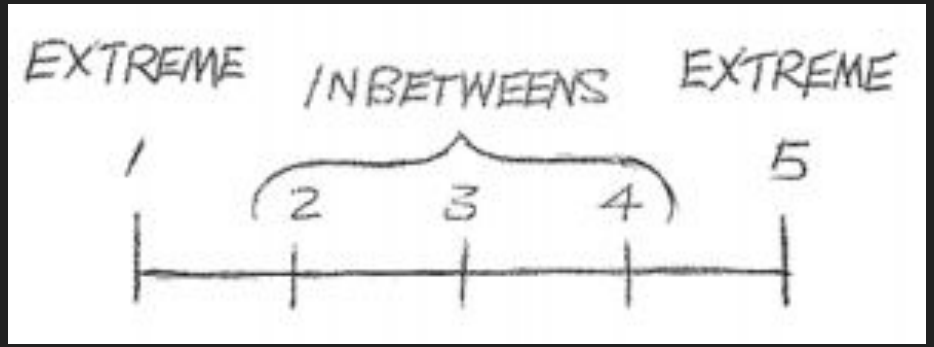
Animation

Source: [comicartfans]

Keyframes



Shuhei Handa, “Little Witch Academia”



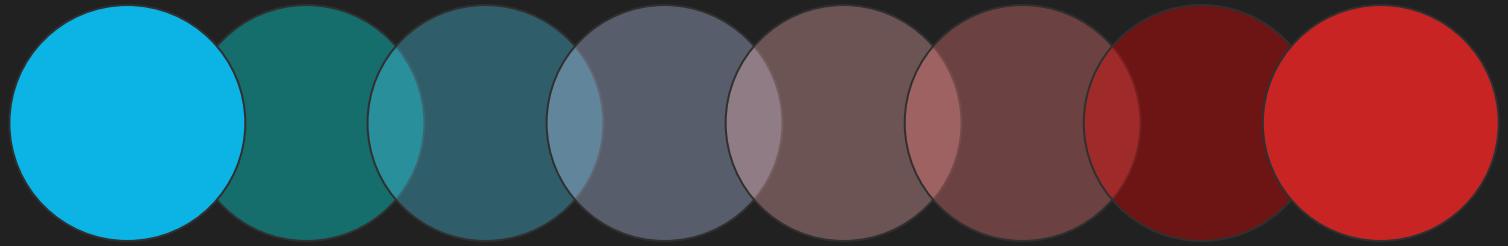
From "The Animator's Survival Kit"
By Richard Williams

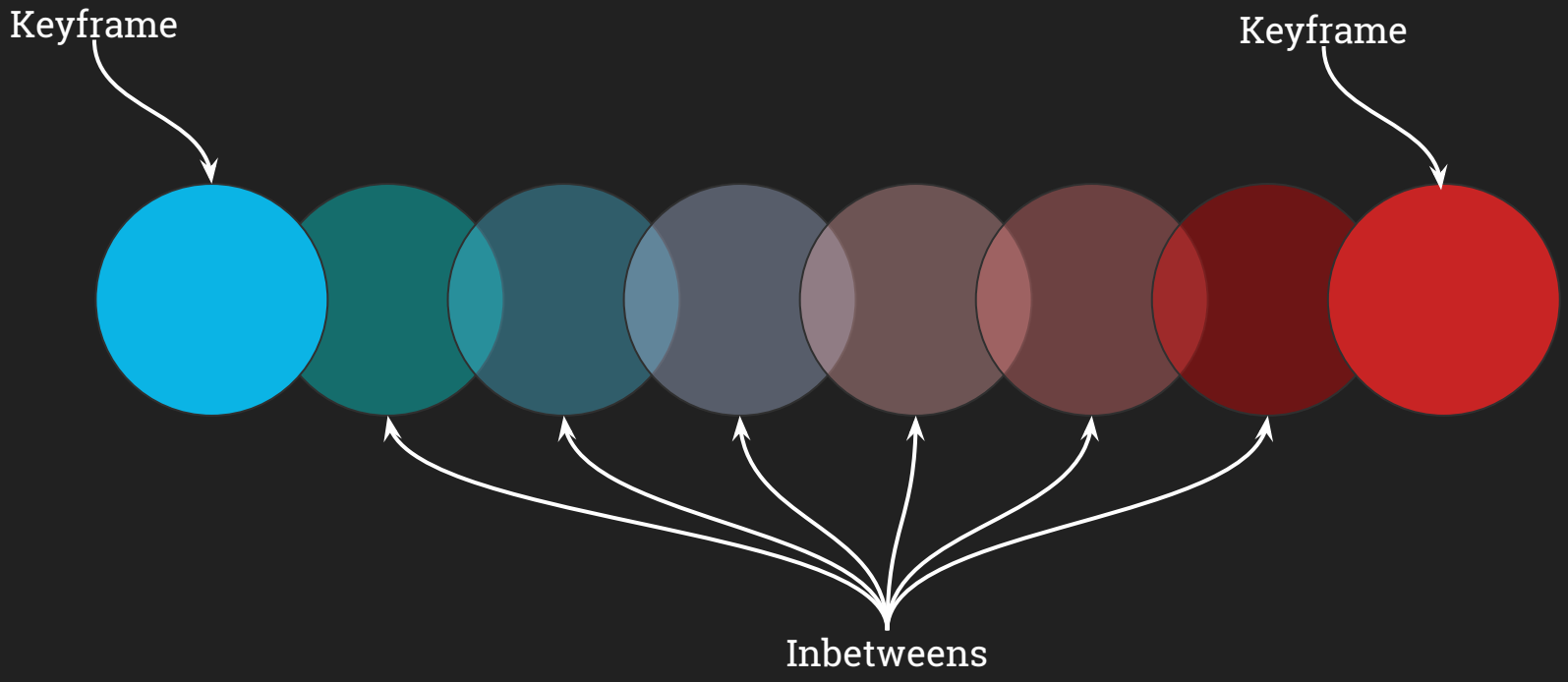
"Inbetweens"

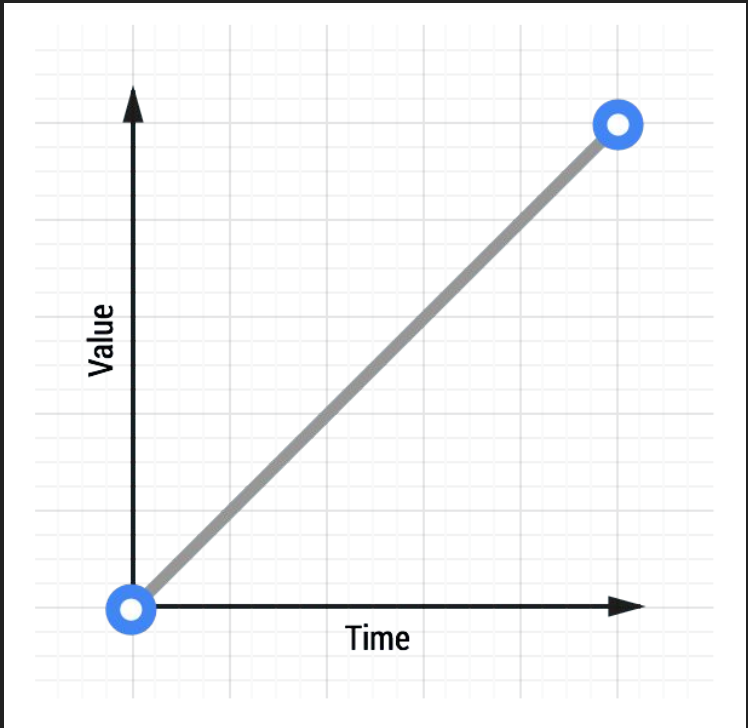
=

"tweens"

CMPM 120



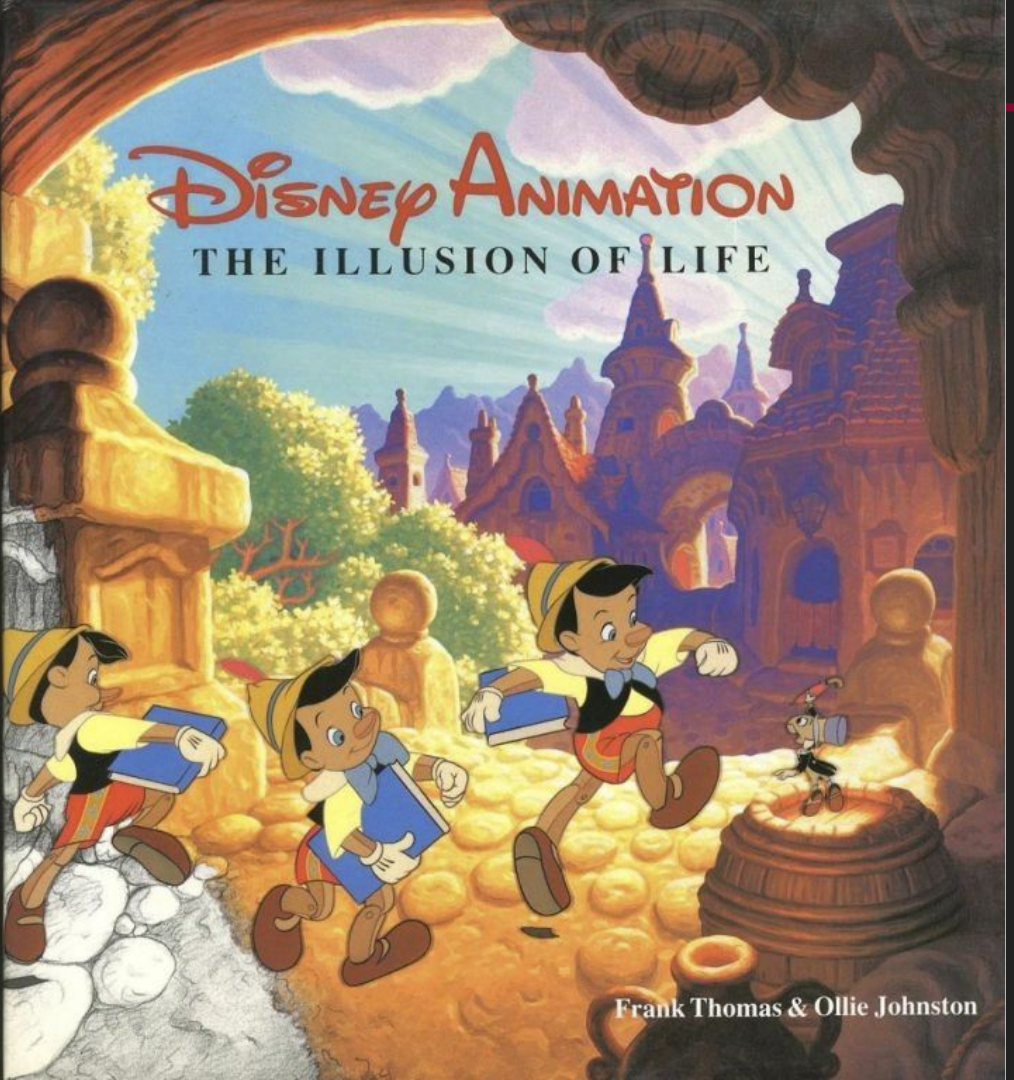

Keyframe
Keyframe
Inbetweens



Point-to-point, linear animation is handy, but it also tends to feel unnatural. Most things in nature tend to accelerate or decelerate as they move.

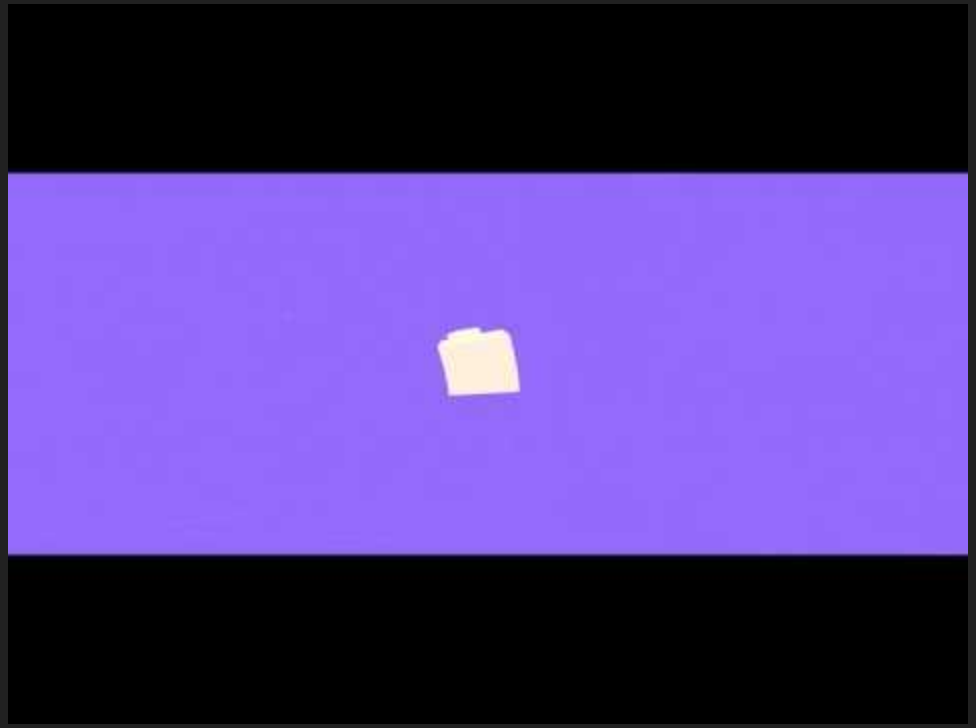
Game developers (especially those influenced by Disney) figured this out early on

You probably noticed that your endless runners felt better with the right amount of inertia to curve the motion

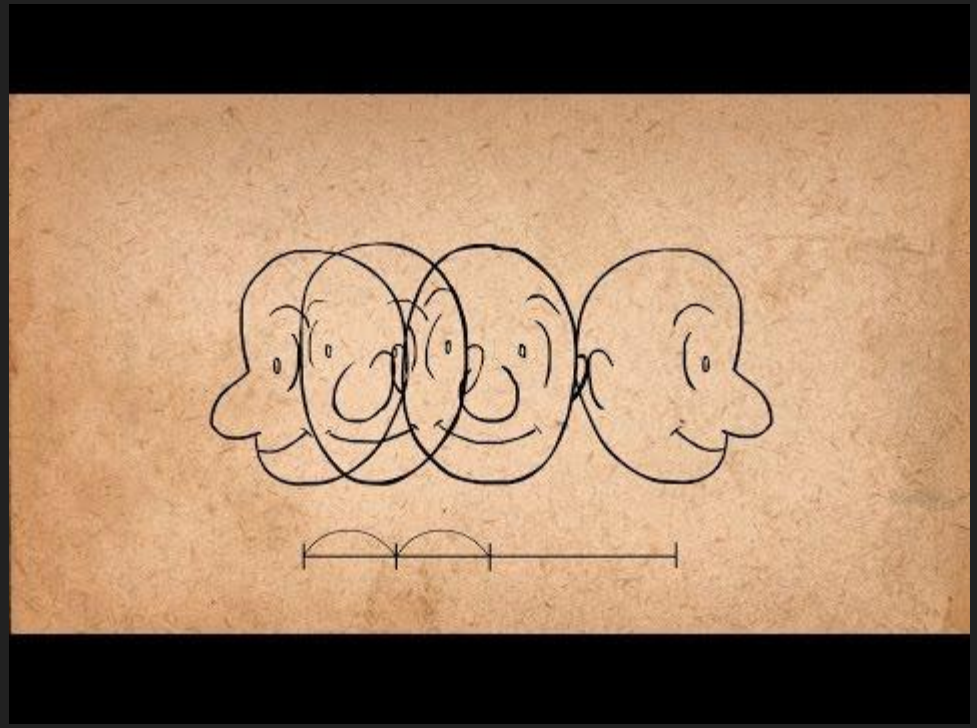


The 12 Principles of Animation

Brown Bag Films [2:20]

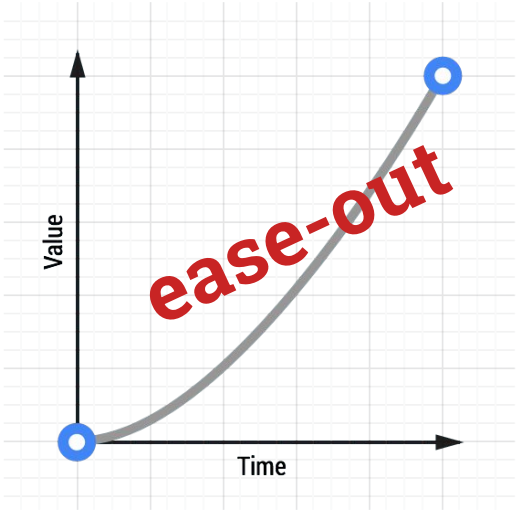


Cento Lodigiani: https://vimeo.com/93206523

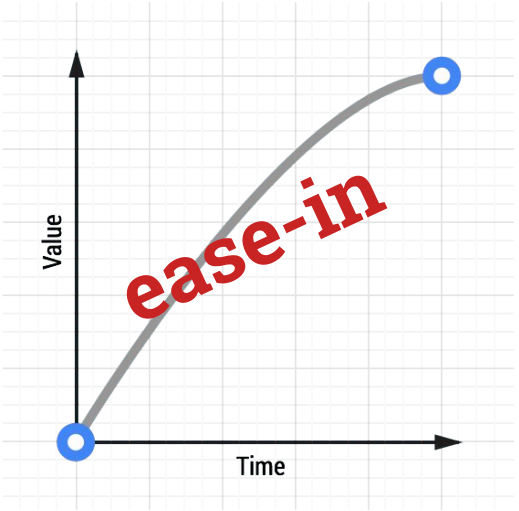


AlanBeckerTutorials

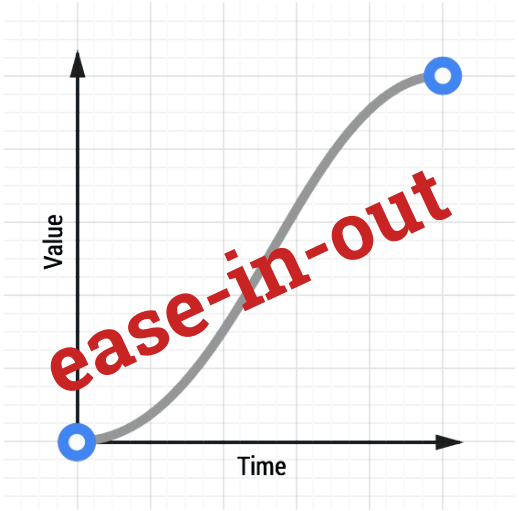
Point-to-point, linear animation is handy, but it also tends to feel unnatural. Most things in nature tend to accelerate or decelerate as they move.



ease-out



ease-in



Ease-in-out

Source: 'The Basics of Easing' and 'Ease-In or Ease-Out?"

Phaser has many tween easing functions

- Linear
- Bounce
- Quadratic
- Cubic
- Quartic
- Quintic
- Sinusoidal
- Exponential
- Circular
- Elastic
- Back

And with the exception of Linear easing, each of these has **In**, **Out**, and **InOut** variants.

If you look at the **`Phaser.Easing`** source code, an easing function is a simple mathematical formula.

For example:

`Cubic.In` returns the **cube** of the value you give it

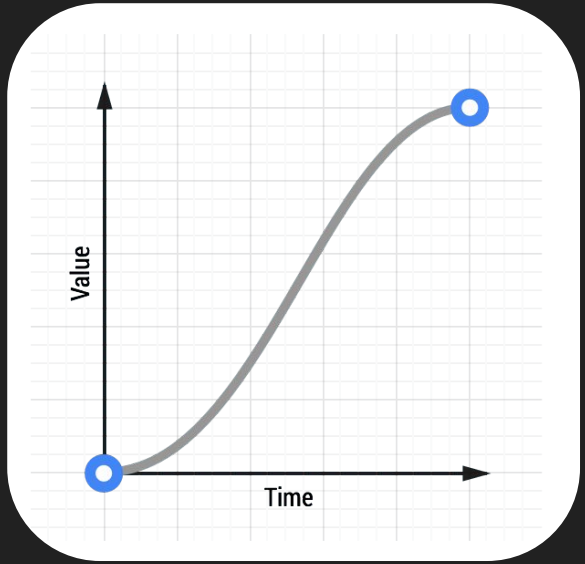
```
91      /**
92      * Cubic easing.
93      *
94      * @class Phaser.Easing.Cubic
95      */
96      Cubic: {
97
98          /**
99          * Cubic ease-in.
100         *
101         * @method Phaser.Easing.Cubic#In
102         * @param {number} k - The value to be tweened.
103         * @returns {number} The tweened value.
104         */
105         In: function (k)
106         {
107
108             return k * k * k;
109
110         },
111
112         /**
113         * Cubic ease-out.
114         *
115         * @method Phaser.Easing.Cubic#Out
116         * @param {number} k - The value to be tweened.
117         * @returns {number} The tweened value.
118         */
119         Out: function (k)
120         {
121
122             return --k * k * k + 1;
123
124         },
```

Between Zero and One

An easing function is basically a way to translate a value between 0 and 1 into a slightly different value

It's also the same as the basic formulas from algebra on a graphing calculator

```
y = x*x*x
```



Look Around You

Lots of things can use curves similar to easing functions:

- ➔ Curve Adjustments in Photoshop
- ➔ Animations
- ➔ Shaders in 3D rendering
- ➔ Procedural generation
- ➔ Particle spawning
- ➔ Gameplay balancing & pacing
- ➔ User interfaces
- ➔ Input handling

Q: What happens if we tween an object but also apply physics?

A: Weird Stuff

"Collisions are highly dependent on the velocity calculations that the physics engine is working with. When you start manually adjusting the position of a physics-enabled object, the velocity becomes almost impossible to correctly calculate, and the separation routines begin failing in odd ways."

lewster32 (in this thread)

More Debugging Tips

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
    a. assert()
    b. Keep a debugging notebook
2. Make debug tools
    a. Quicker feedback is better
    b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
    a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
    a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

Walk through your code step by step, explaining to yourself what is supposed to happen

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
    a. assert()
    b. Keep a debugging notebook
2. Make debug tools
    a. Quicker feedback is better
    b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
    a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
    a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
    a. assert()
    b. Keep a debugging notebook
2. Make debug tools
    a. Quicker feedback is better
    b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
    a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
    a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
    a. assert()
    b. Keep a debugging notebook
2. Make debug tools
    a. Quicker feedback is better
    b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
    a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
    a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
    a. assert()
    b. Keep a debugging notebook
2. Make debug tools
    a. Quicker feedback is better
    b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
    a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
    a. Faster to print an array as a string than to individually print the contents

AABB characters and slopes

An example of a real-world physics-and-debugging problem in a game with 2D physics like yours

https://twitter.com/eevee/status/1133248372624613376