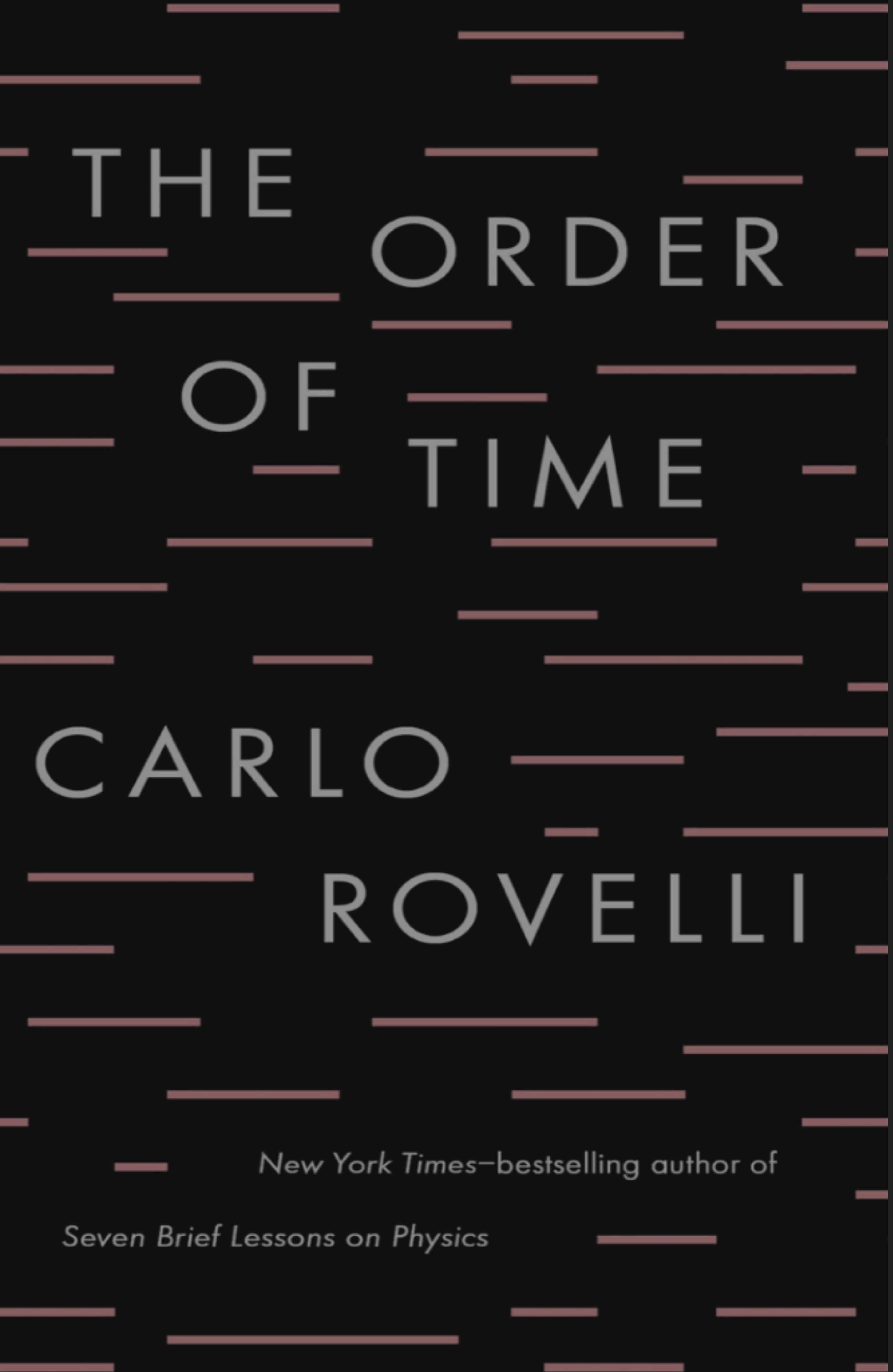
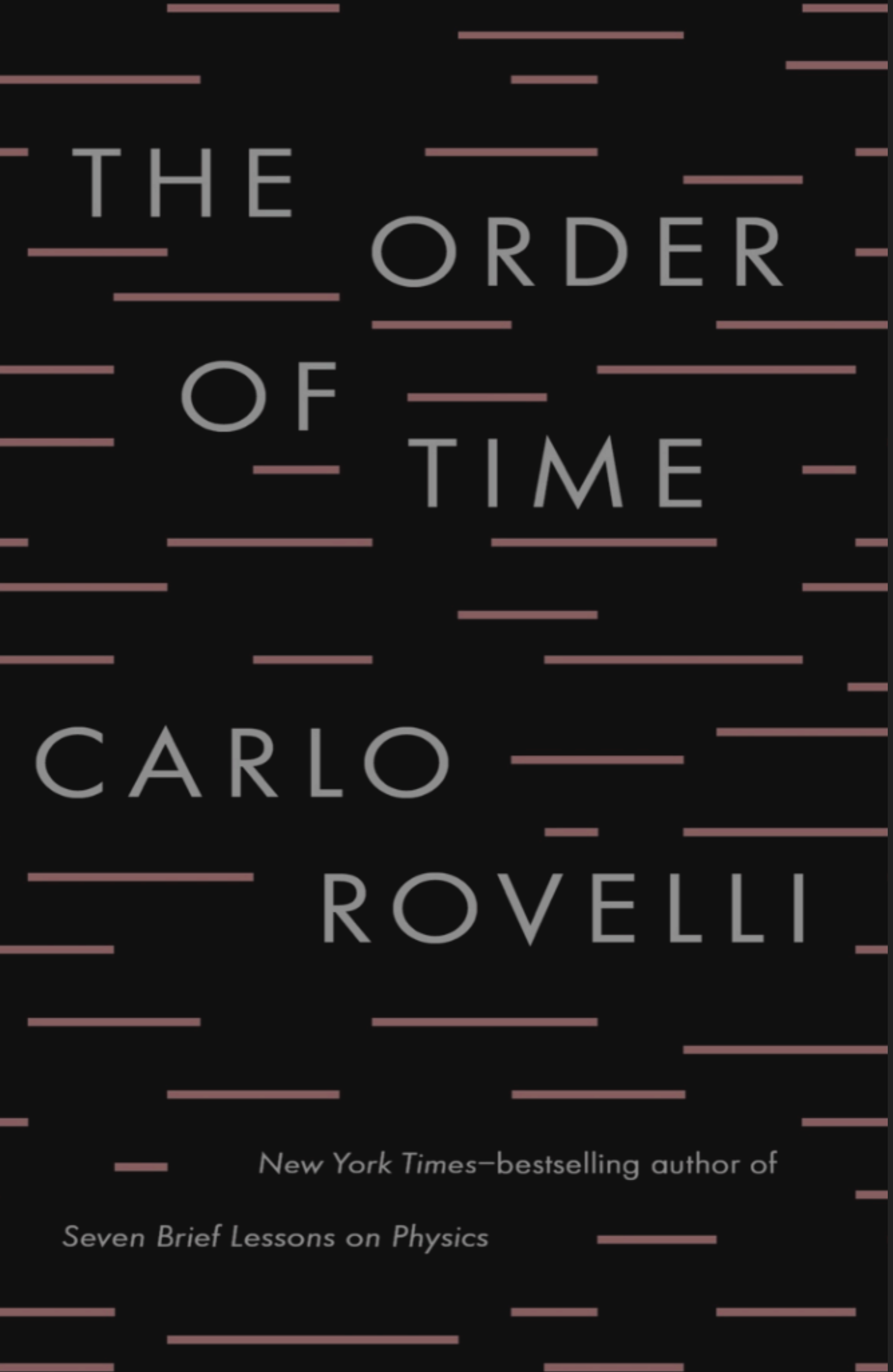
EXISTING IN TIME

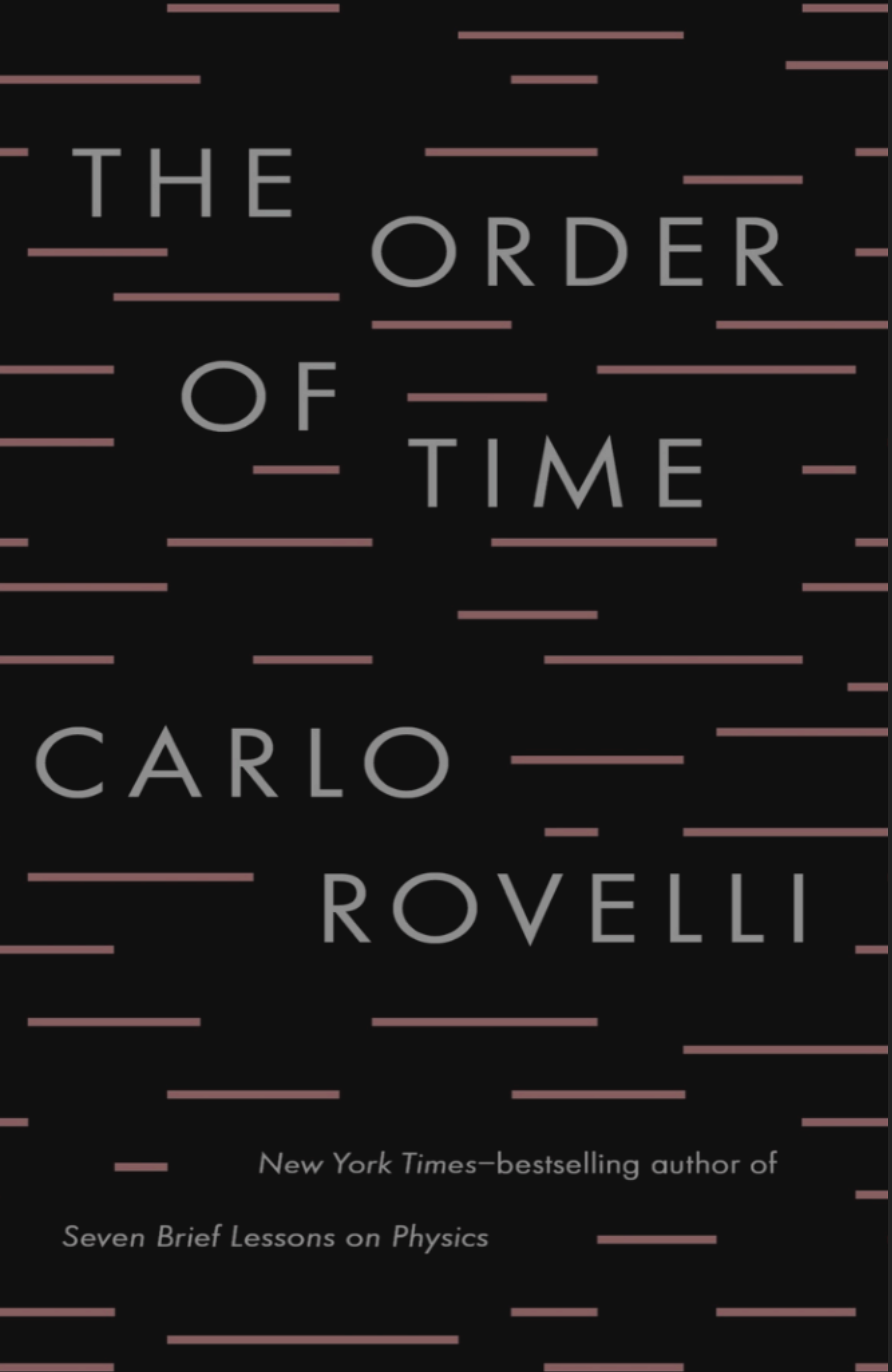
Aka Time Code



'The difference between things and events is that things persist in time; events have a limited duration. A stone is a prototypical "thing": we can ask ourselves where it will be tomorrow. Conversely, a kiss is an "event." It makes no sense to ask where the kiss will be tomorrow. The world is made up of a network of kisses, not of stones.'



'On closer inspection, even the things that are most "thinglike" are nothing more than long events. The hardest stone . . . is in reality a complex vibration of quantum fields, a momentary interaction of forces, a process that for a brief moment [eons] manages to keep its shape.'



‘On closer inspection, even the things that are most “thinglike” are nothing more than long events. The hardest stone, in the light of what we have learned from chemistry, physics, from mineralogy, from geology, from psychology, is in reality a complex vibration of quantum fields, a momentary interaction of forces, a process that for a brief moment [eons] manages to keep its shape, to hold itself in equilibrium before disintegrating into dust, a brief chapter in the history of interactions between the elements of the planet, a trace of Neolithic humanity, a weapon used by a gang of kids, an example in a book about time, a metaphor for an ontology, a part of a segmentation of the world that depends more on how our bodies are structured to perceive than on the object of perception – and, gradually, an intricate knot in that cosmis game of mirrors that constitutes reality. The world is not so much made up of stones as of fleeting sounds, or of waves moving through the sea. ’


OFF

Soundcraft Vi6

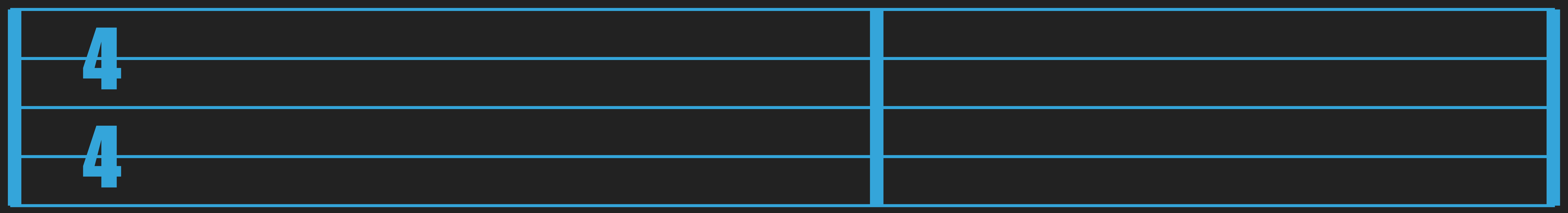

SEASENSE
Marine / Sport
HORN
Recommended for boats up to 12 meters / 39.4 feet
Ideal for all sporting events
KEEP OUT OF REACH OF CHILDREN.

CODE

More situated in time than linear media or static artifacts
Temporal and meta-temporal medium

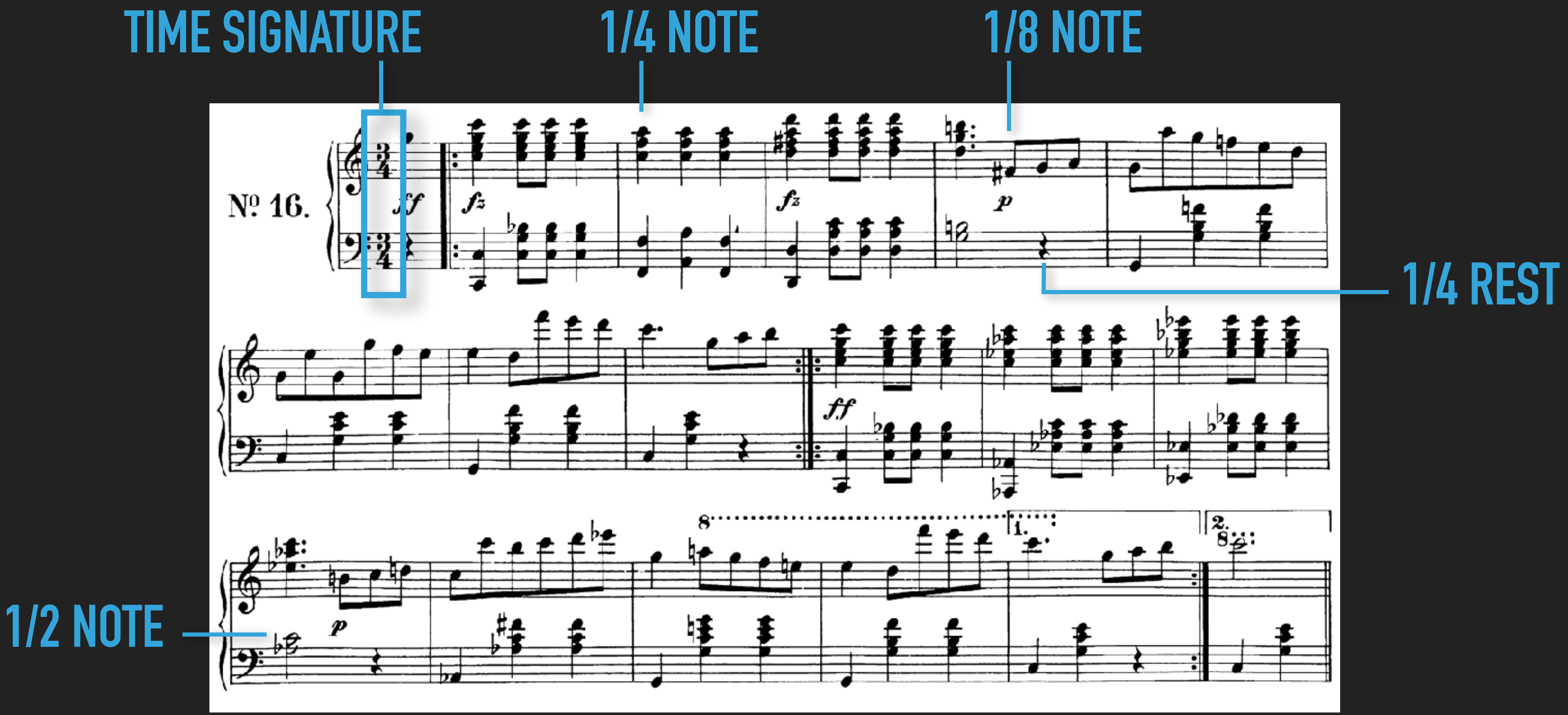
~~CODE~~ MUSIC

Situated in time, with loops and branches, but not self-referential



~~CODE~~ MUSIC

Symbols for representing time

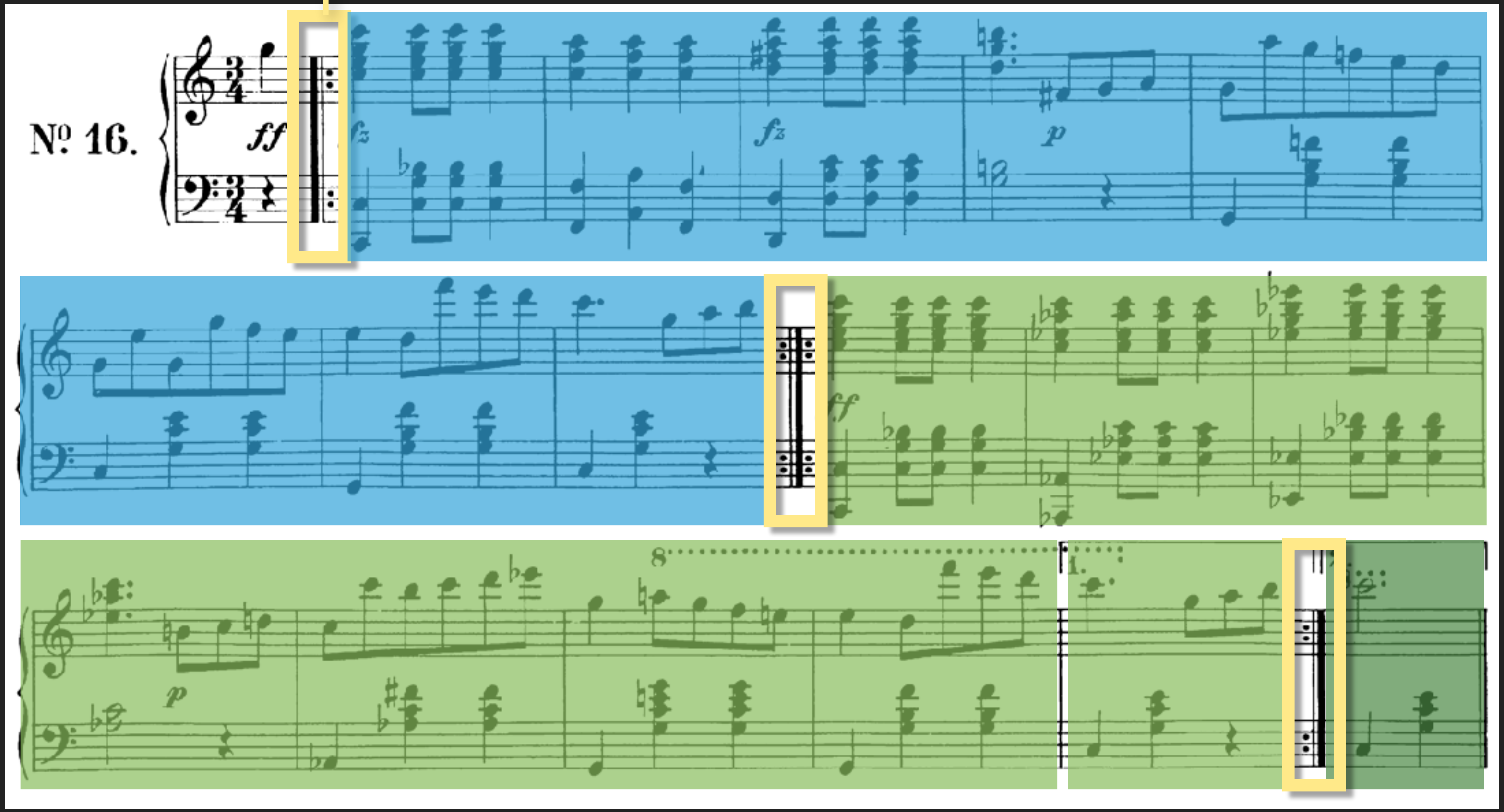


~~CODE~~ MUSIC

Symbols for structuring execution

LOOP DELIMITER SYMBOL

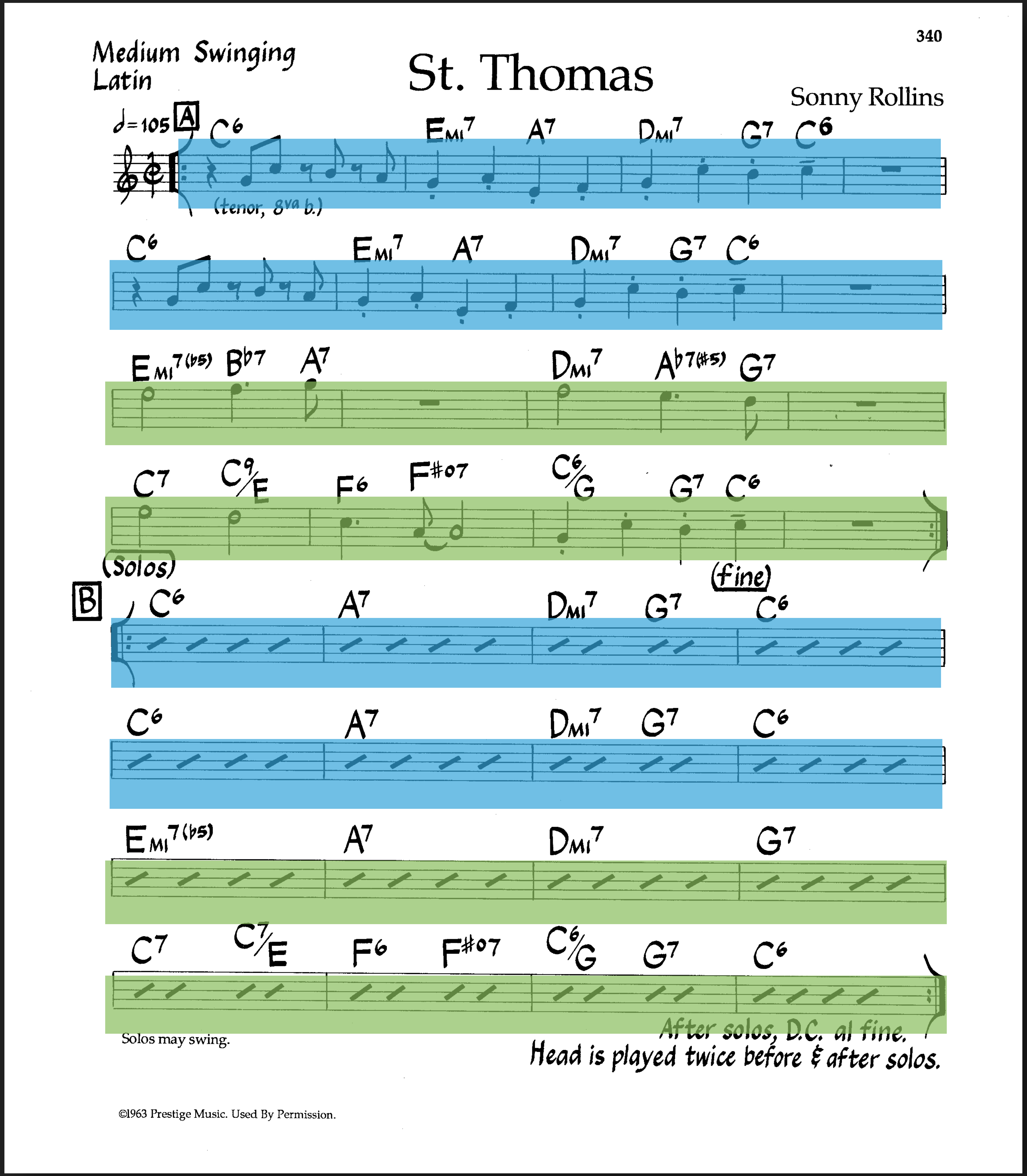
REPEAT FIRST SECTION ONCE

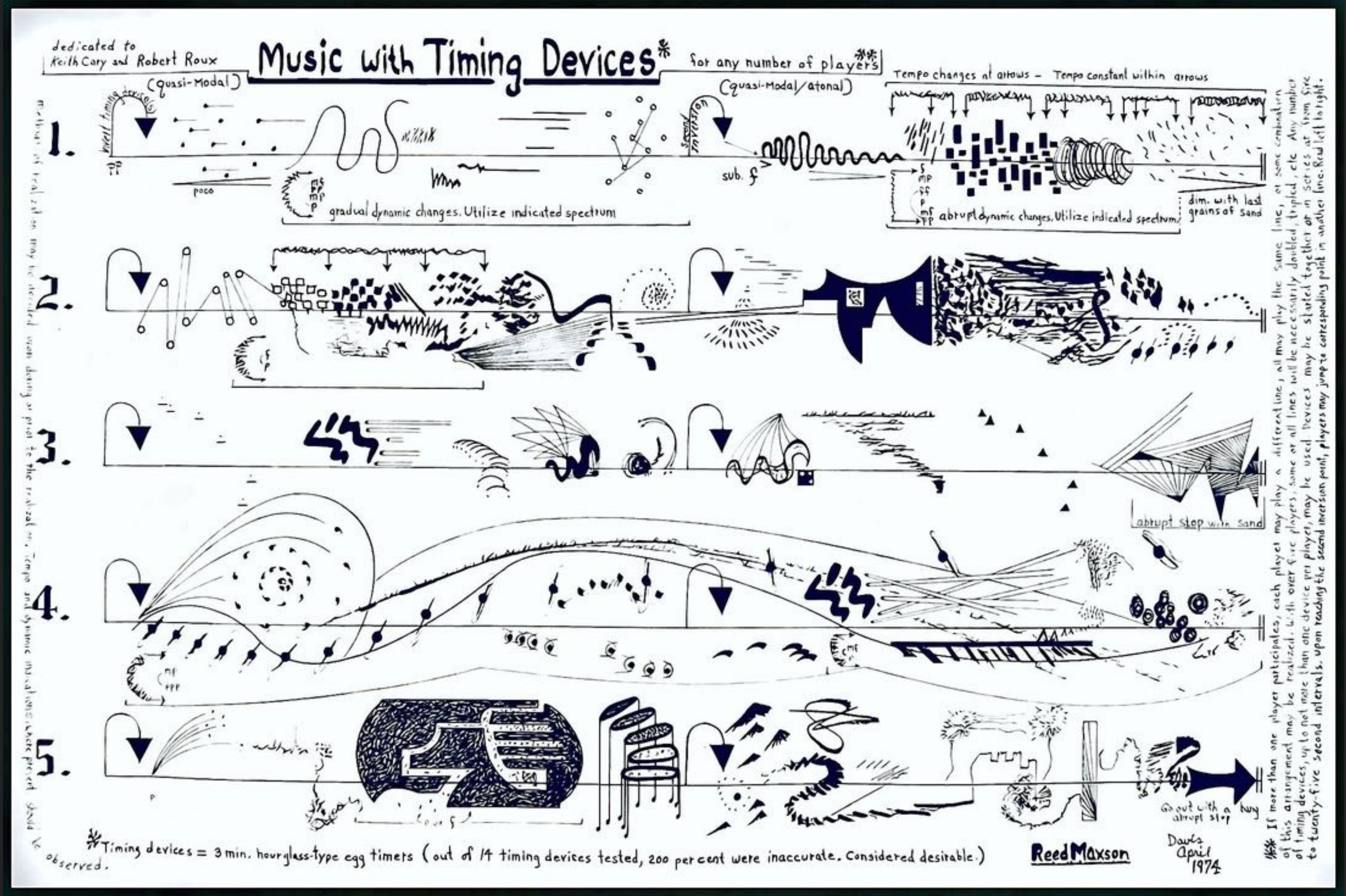


REPEAT SECOND SECTION, USE ALTERNATE ENDING SECOND TIME

~~CODE~~ MUSIC

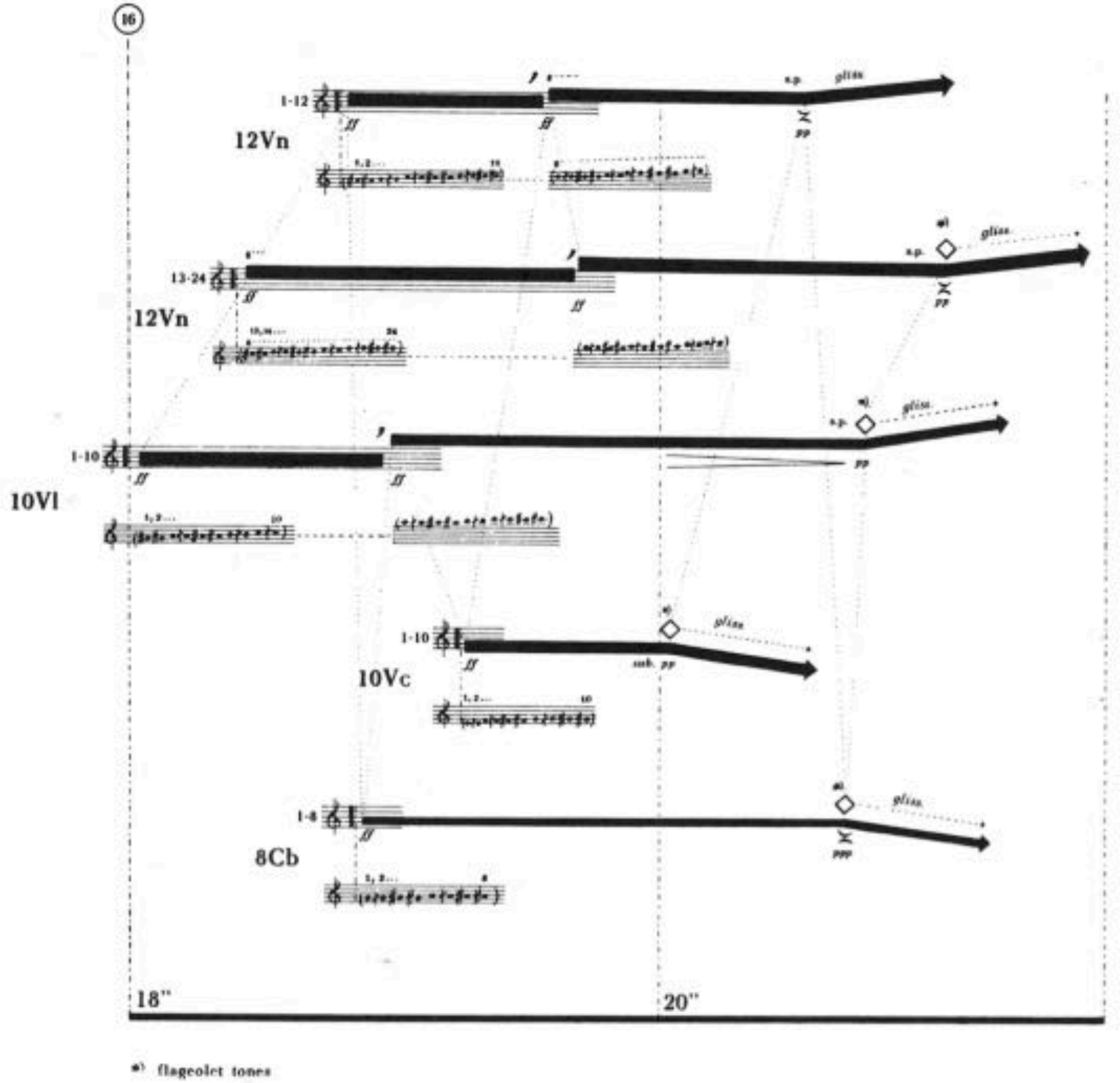
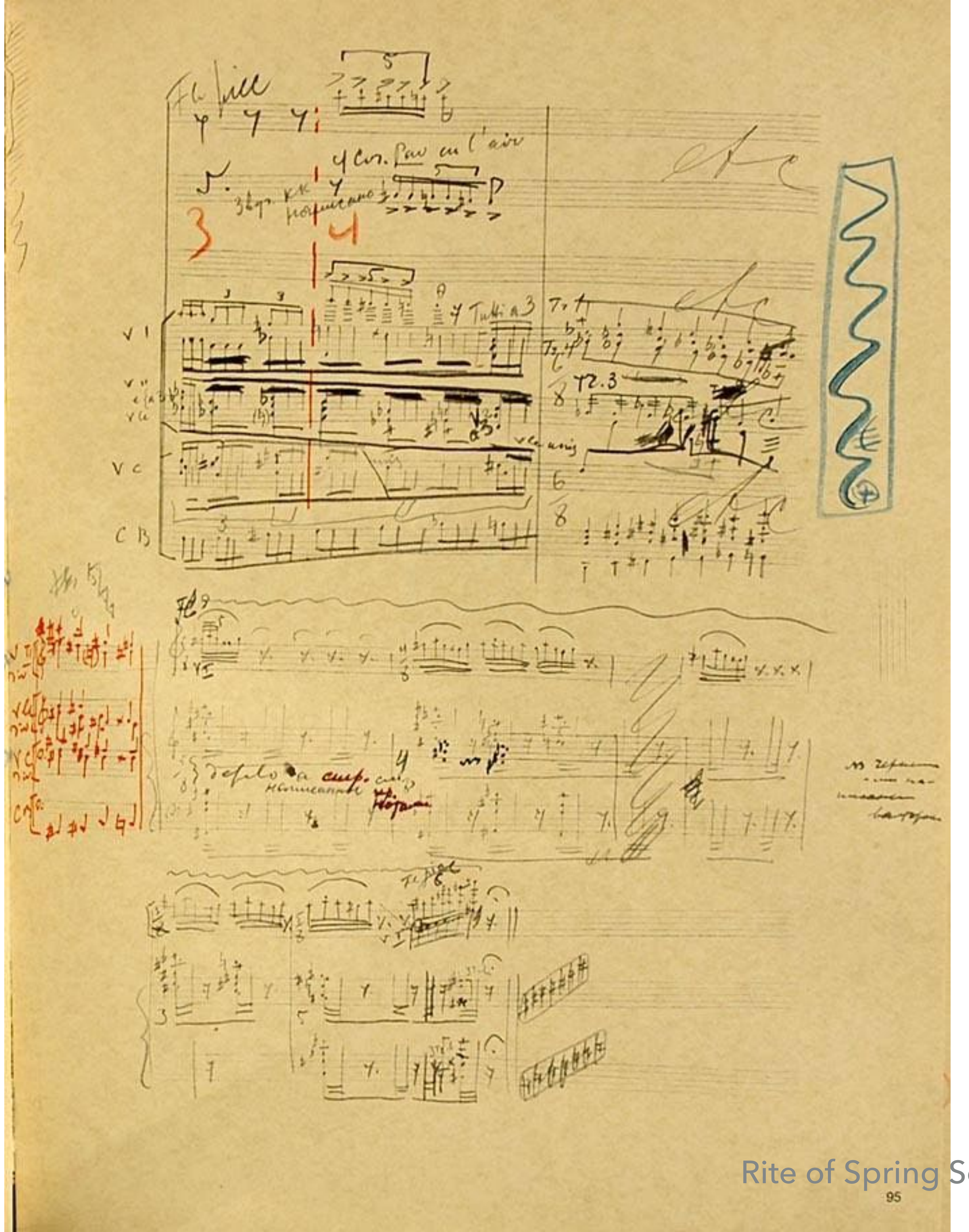
Simple outline can be decoded in sophisticated way by specially-trained agents aka 'musicians'



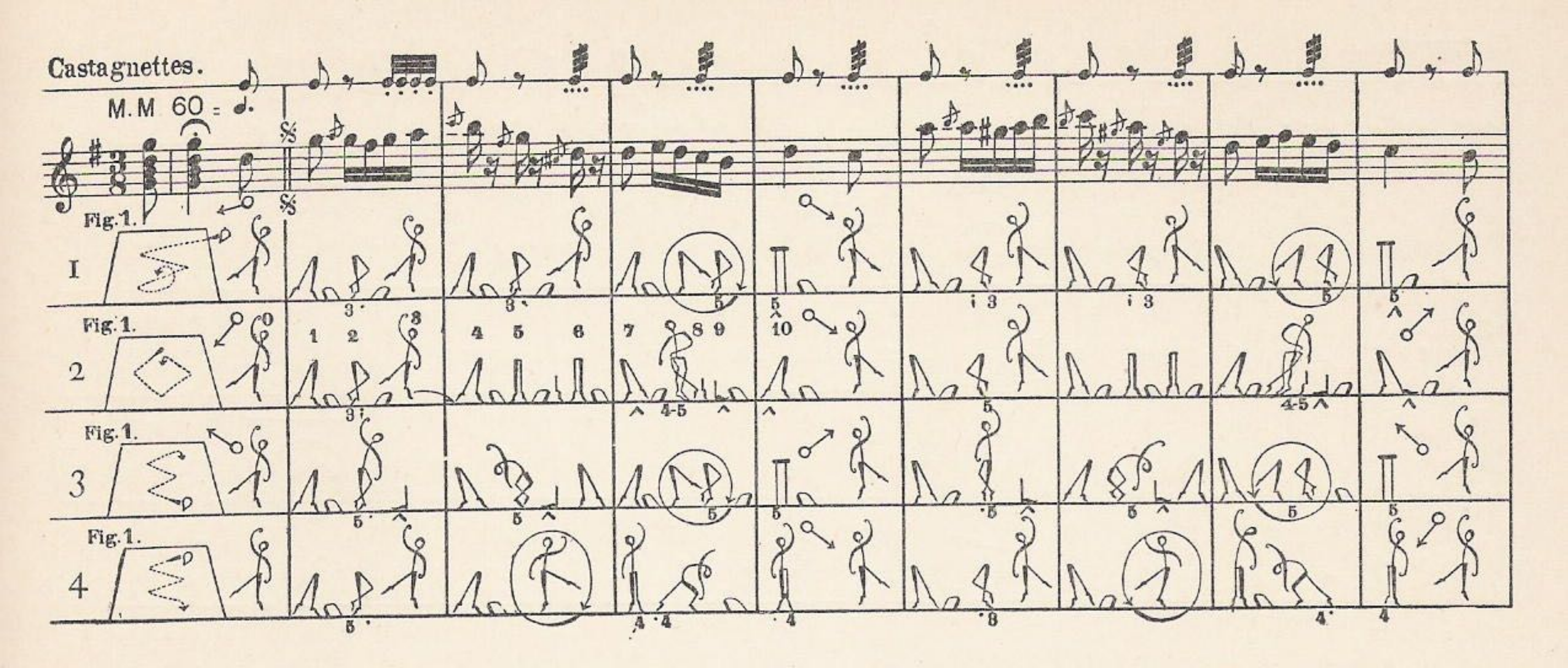


https://www.reedmaxson.com/graphic-scores.html

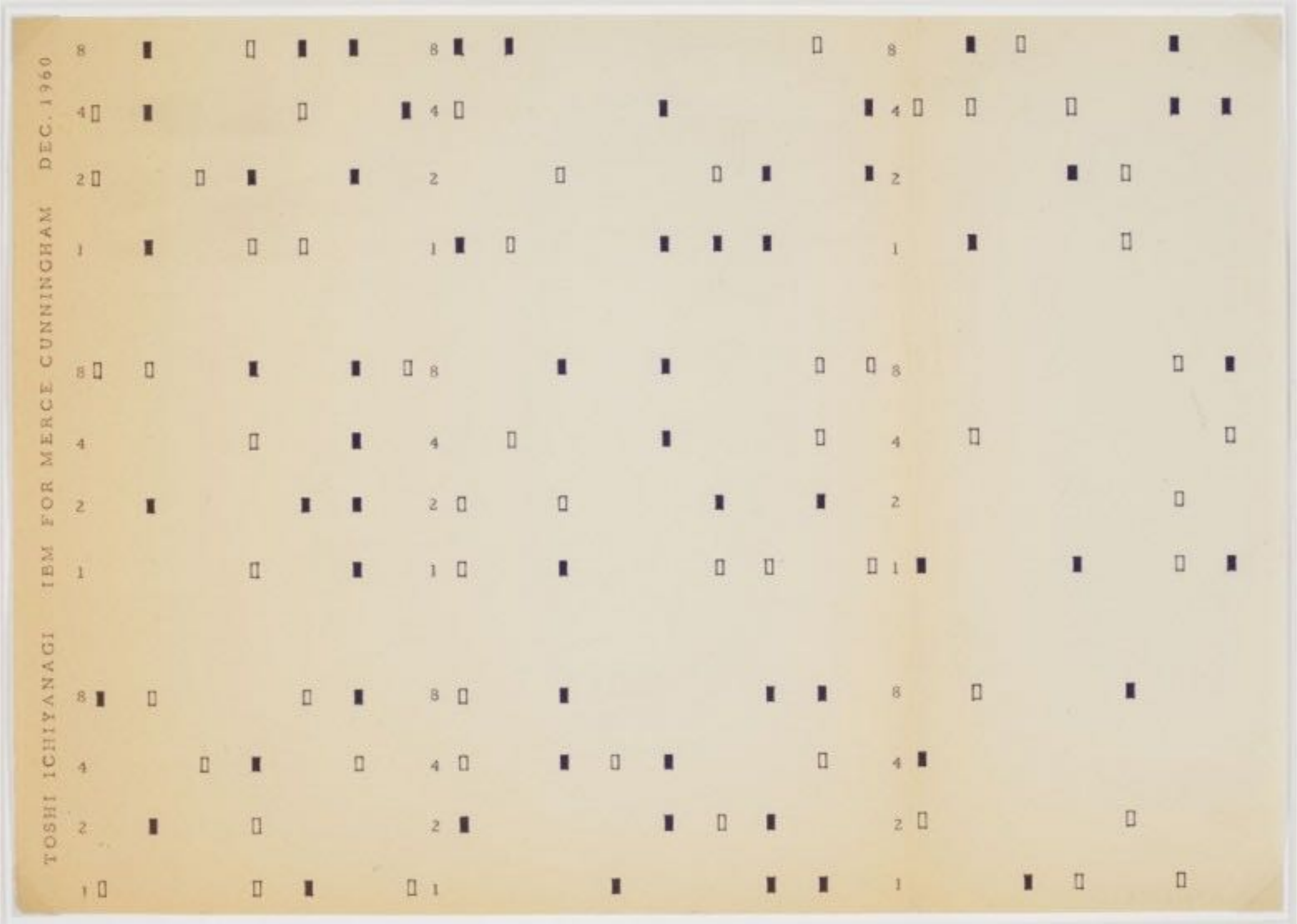
Krzysztof Penderecki: *Threnody for the Victims of Hiroshima* (1960)



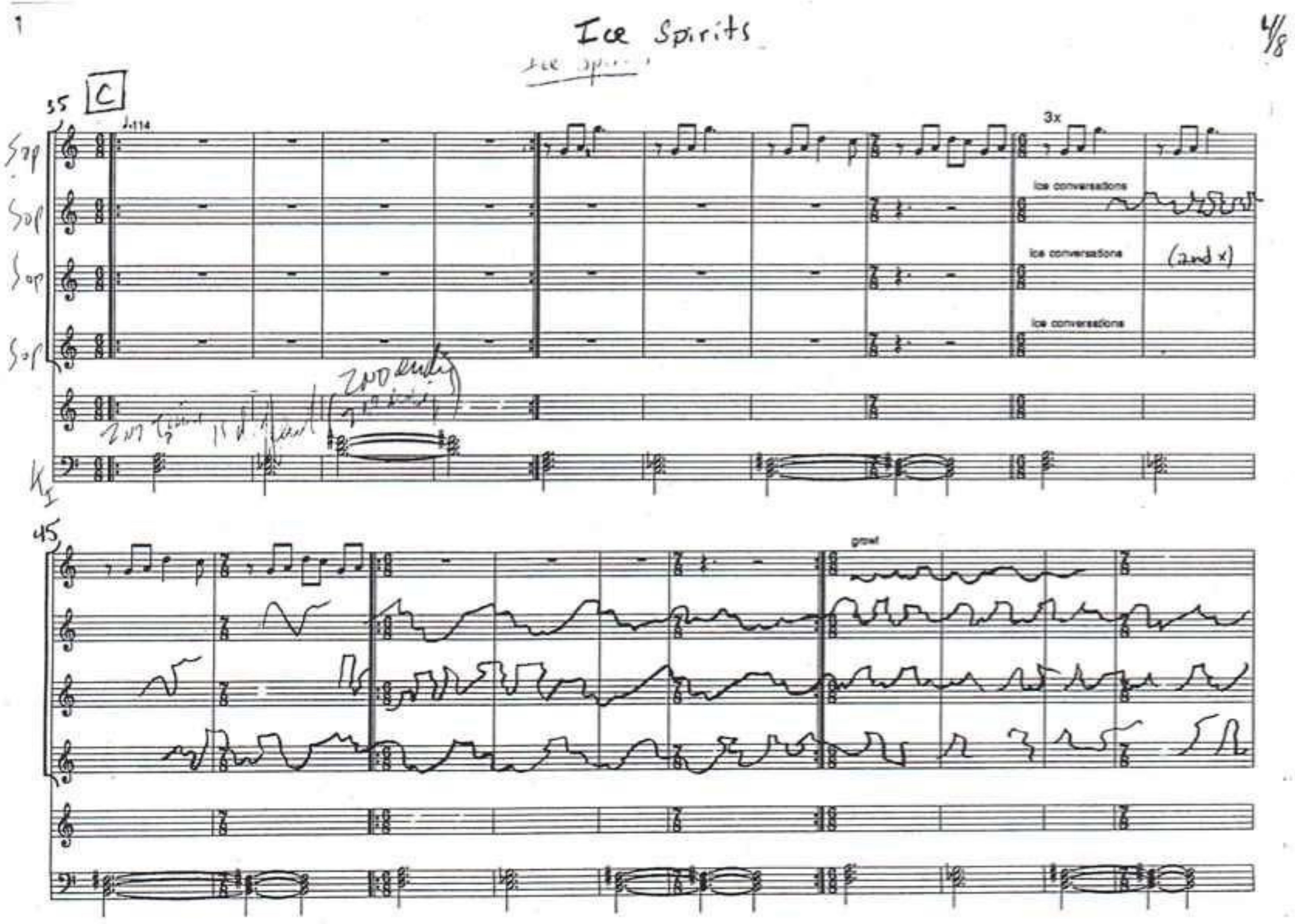
Rite of Spring Score, Igor Stravinsky

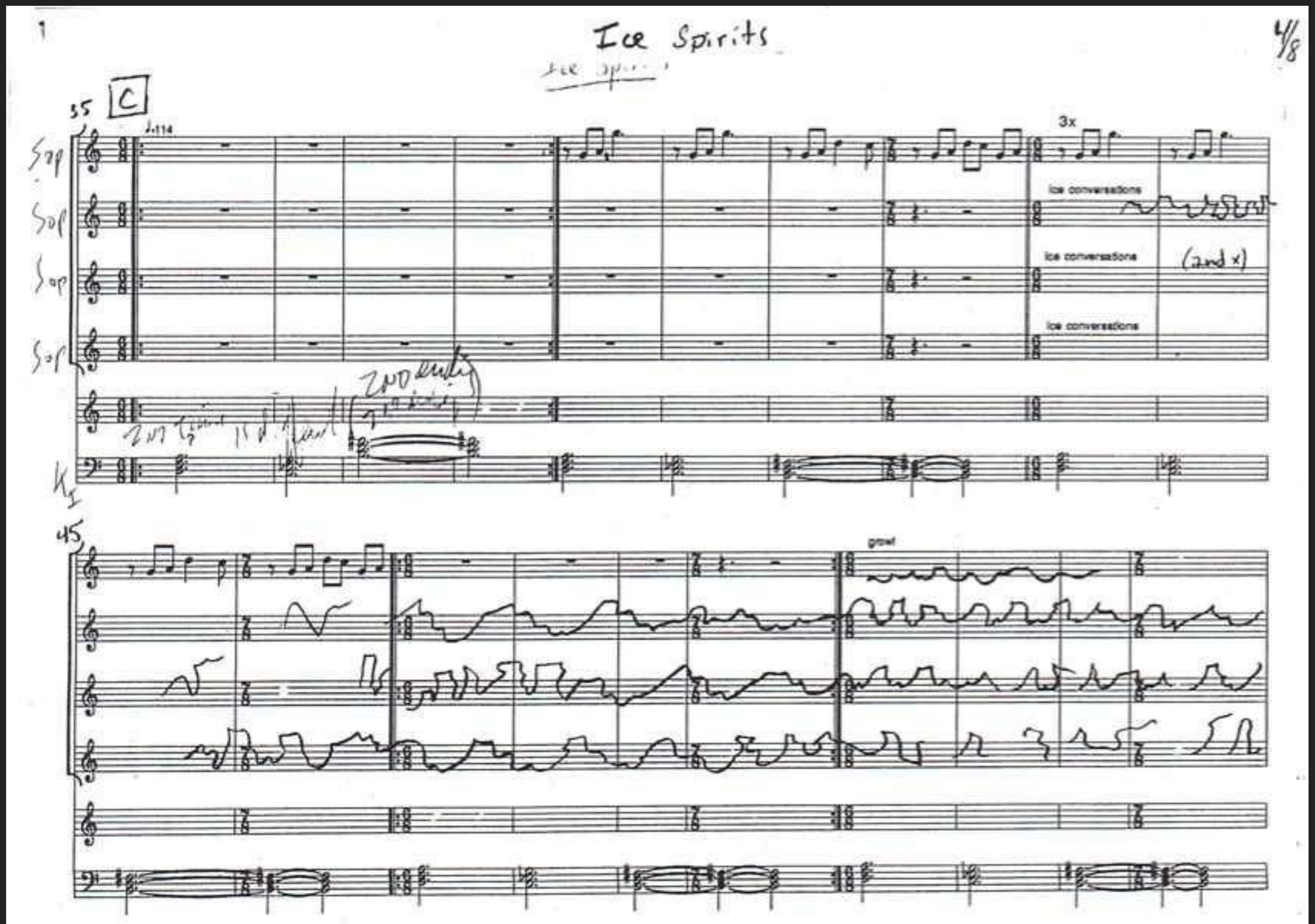


La Cachucha, by Friedrich Albert Zorn (wikipedia)



https://www.moma.org/explore/inside_out/2012/12/21/exhibiting-fluxus-keeping-score-in-tokyo-1955-1970-a-new-avant-garde/

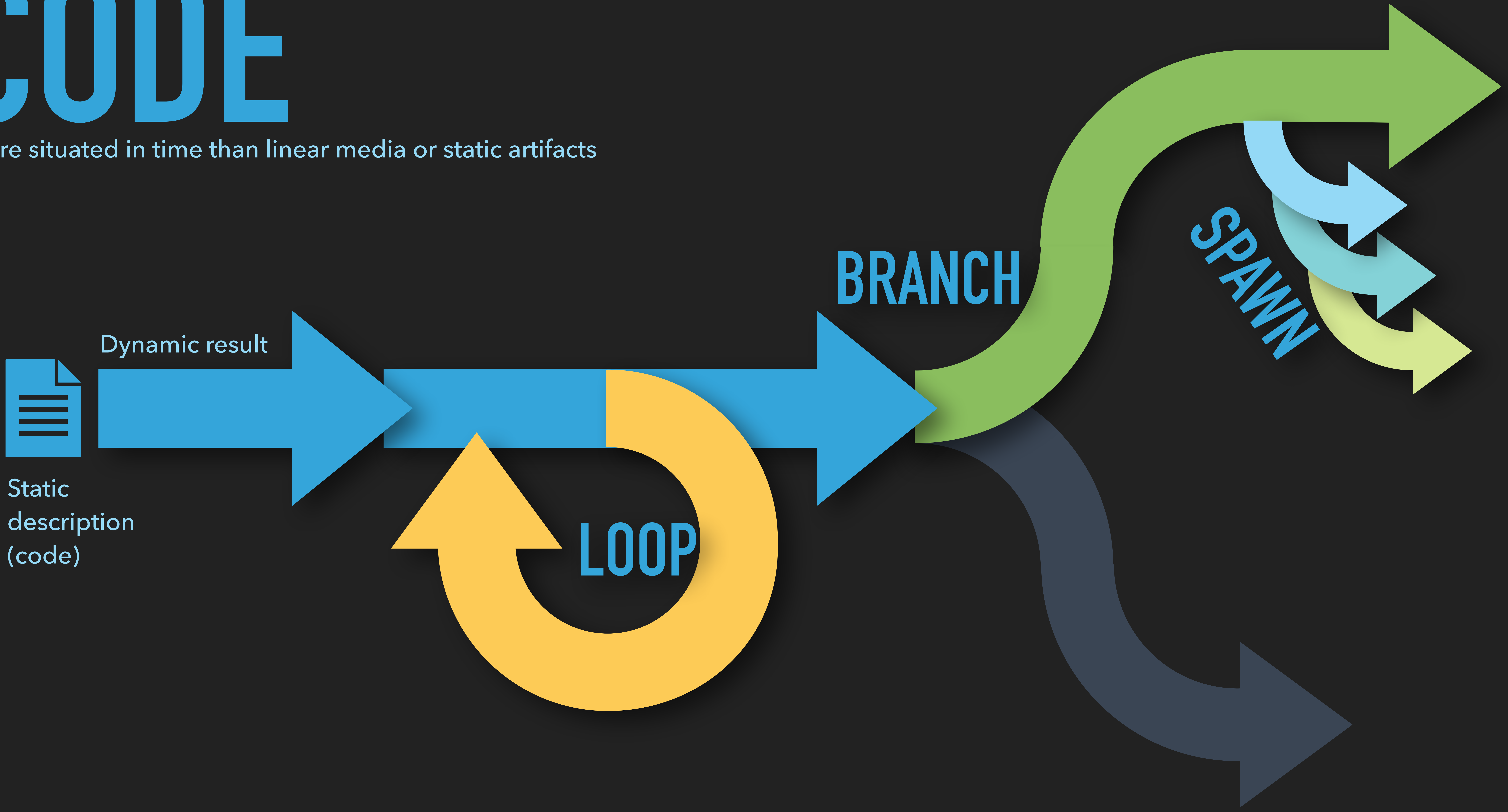
482 × 283
5.4525
5.5025
POUR WATER FROM ONE RECEPTACLE TO ANOTHER
SIREN-WHISTLE
6.3025
POUR WATER
RYOANJI
JOHN CAGE
ORGAN²/ASLSP
4'33"
SOLO FOR VOICE 95

Score of Ice Spirits, Meredith Monk



These things can make music, but neither of them is music

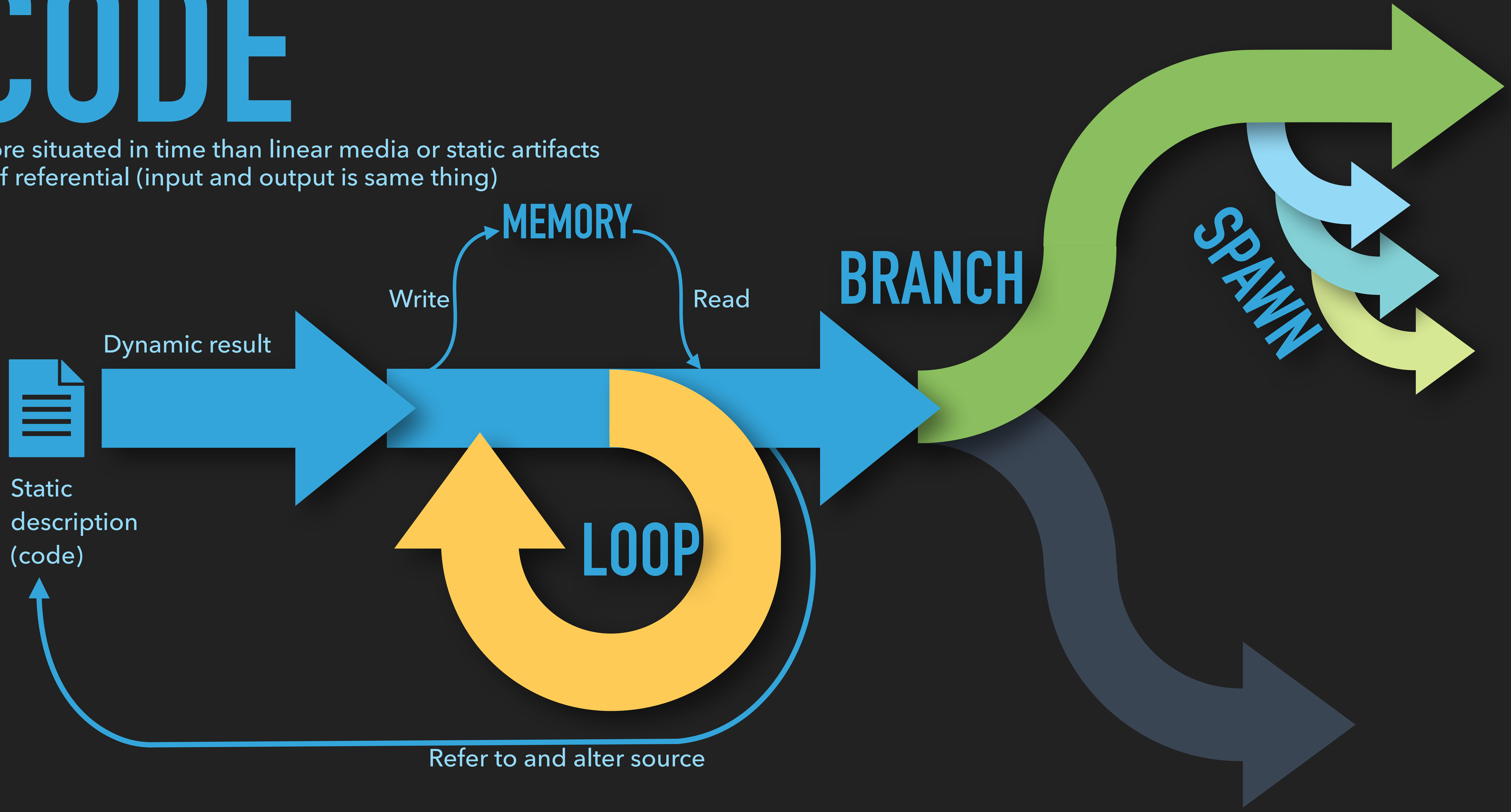
CODE

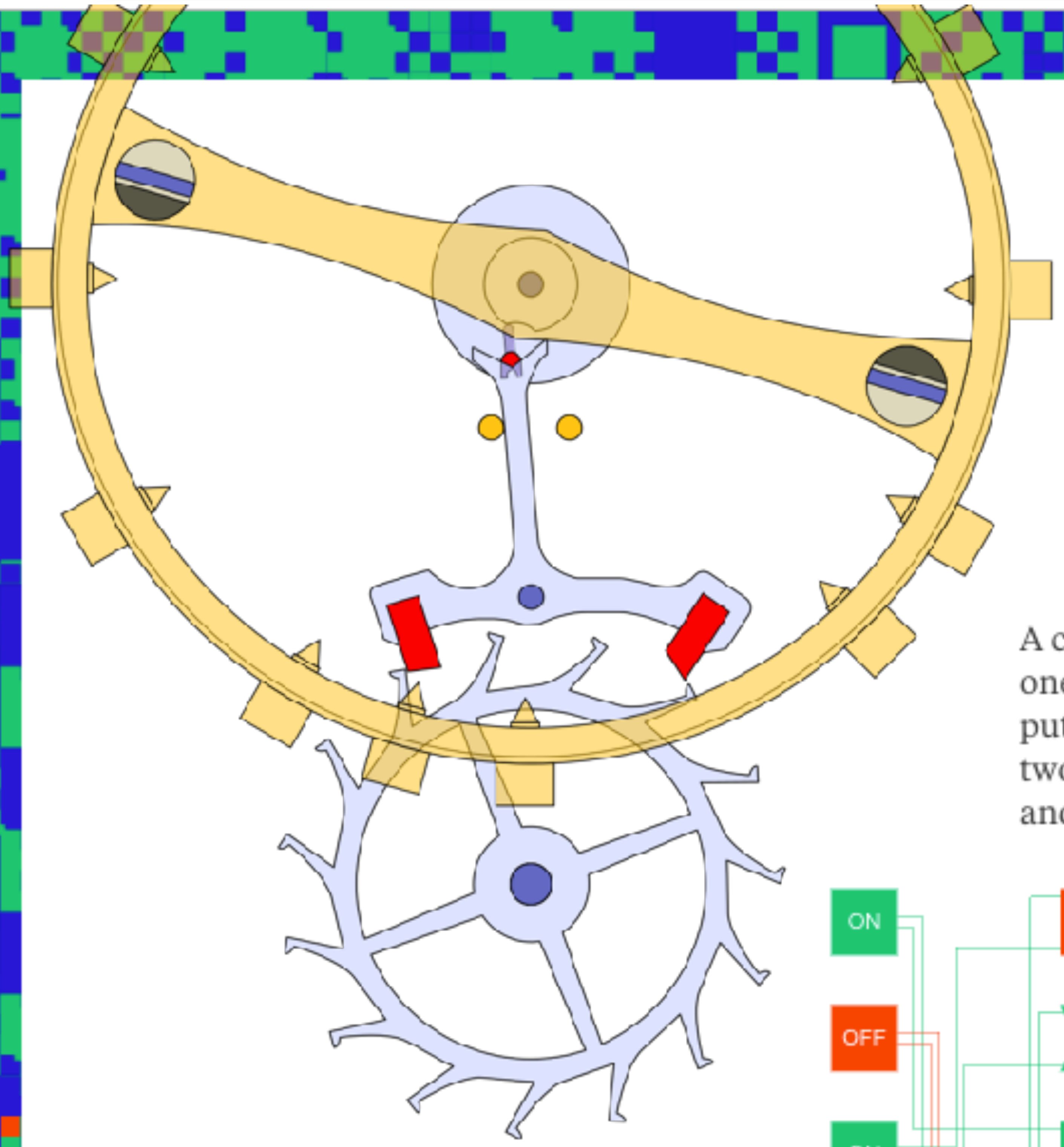
More situated in time than linear media or static artifacts



CODE

More situated in time than linear media or static artifacts
Self referential (input and output is same thing)

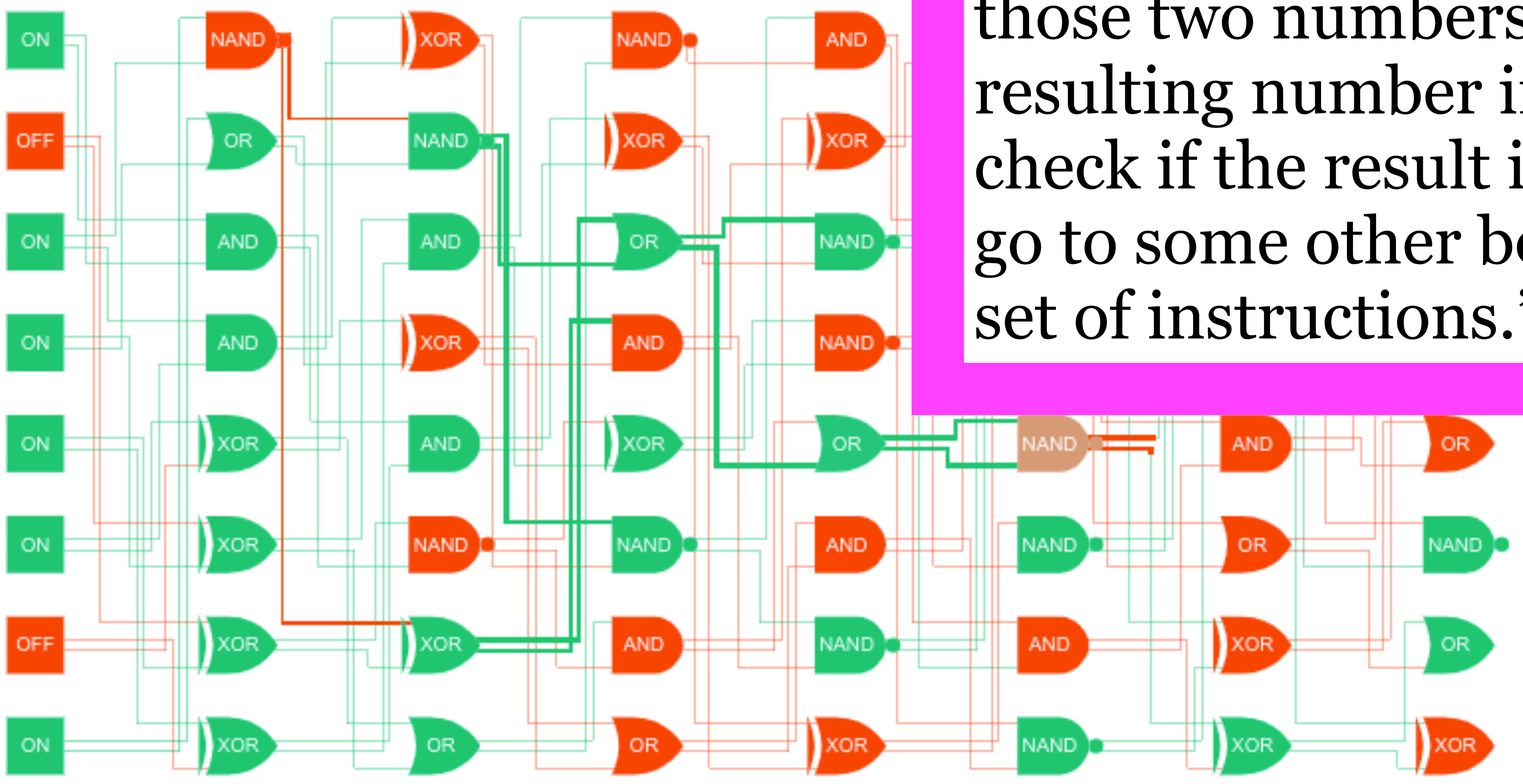


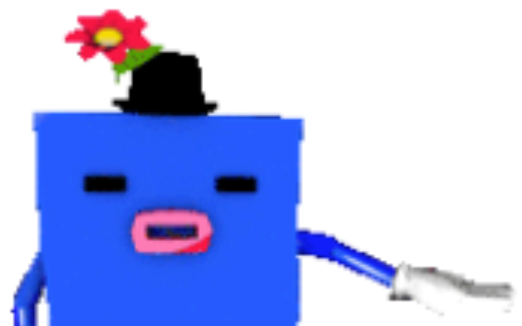


2

Let's Be

A computer is a clock with benefits. They all work the sa
one step at a time: Tick, take a number and put it in box
put it in box two. Tick, *operate* (an operation might be a
two numbers and put the resulting number in box one. T
and if it is, go to some other box and follow a new set of



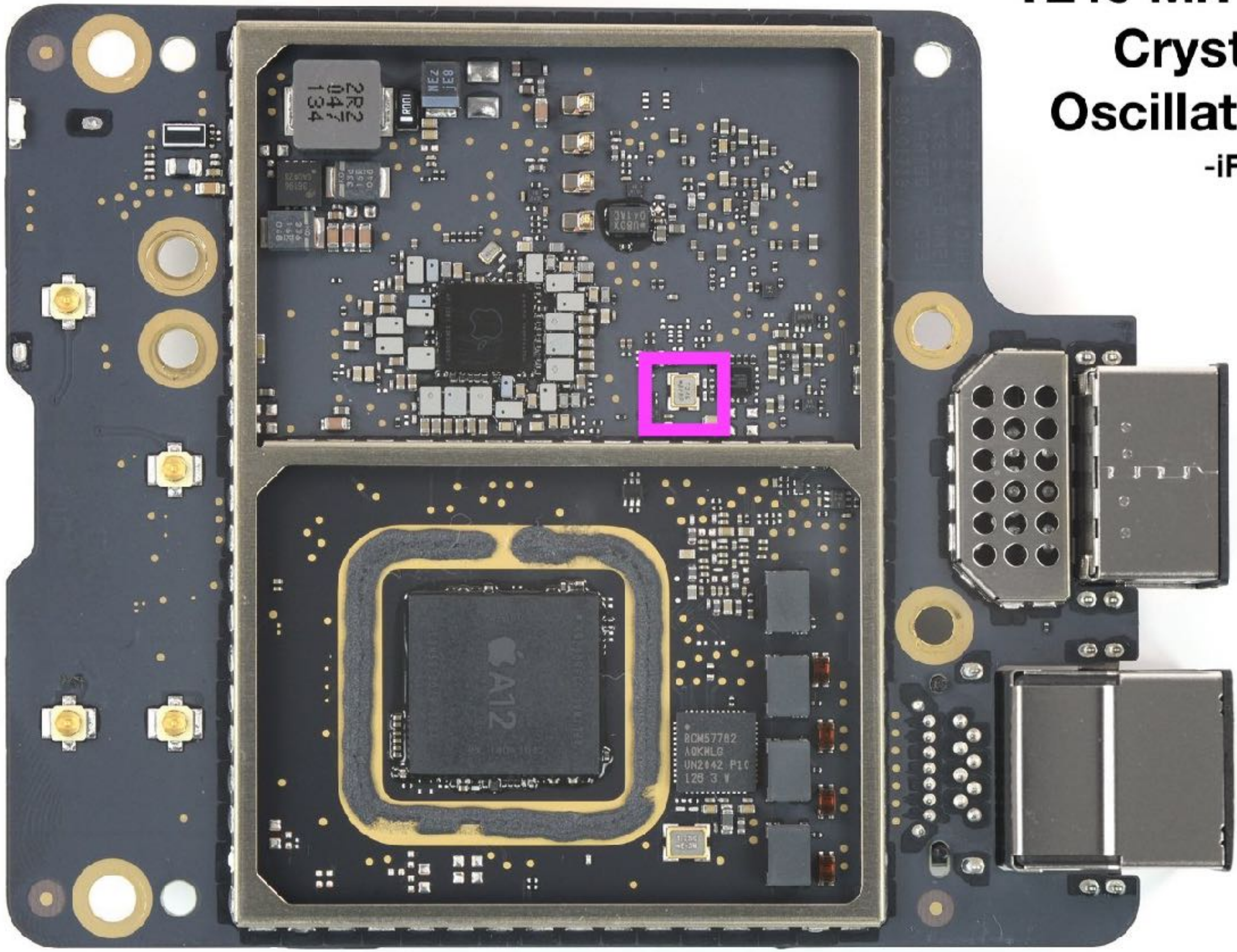


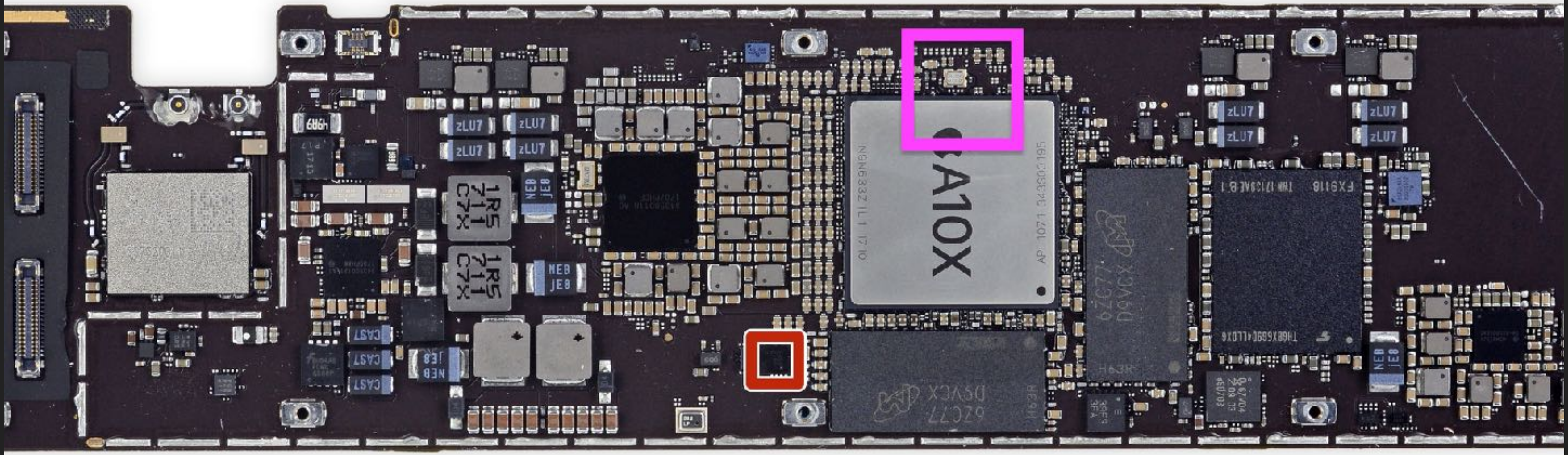
This is simulated circuitry that's computing as you watch. The switches on the left turn the current on and off at random, and the logic gates direct the flow of the current. Click the boxes to change the circuits. Enough of these can compute anything computable.

"A computer is a clock with benefits. They all work the same, doing second-grade math, one step at a time: Tick, take a number and put it in box one. Tick, take another number, put it in box two. Tick, *operate* (an operation might be addition or subtraction) on those two numbers and put the resulting number in box one. Tick, check if the result is zero, and if it is, go to some other box and follow a new set of instructions."

Crystal Oscillator

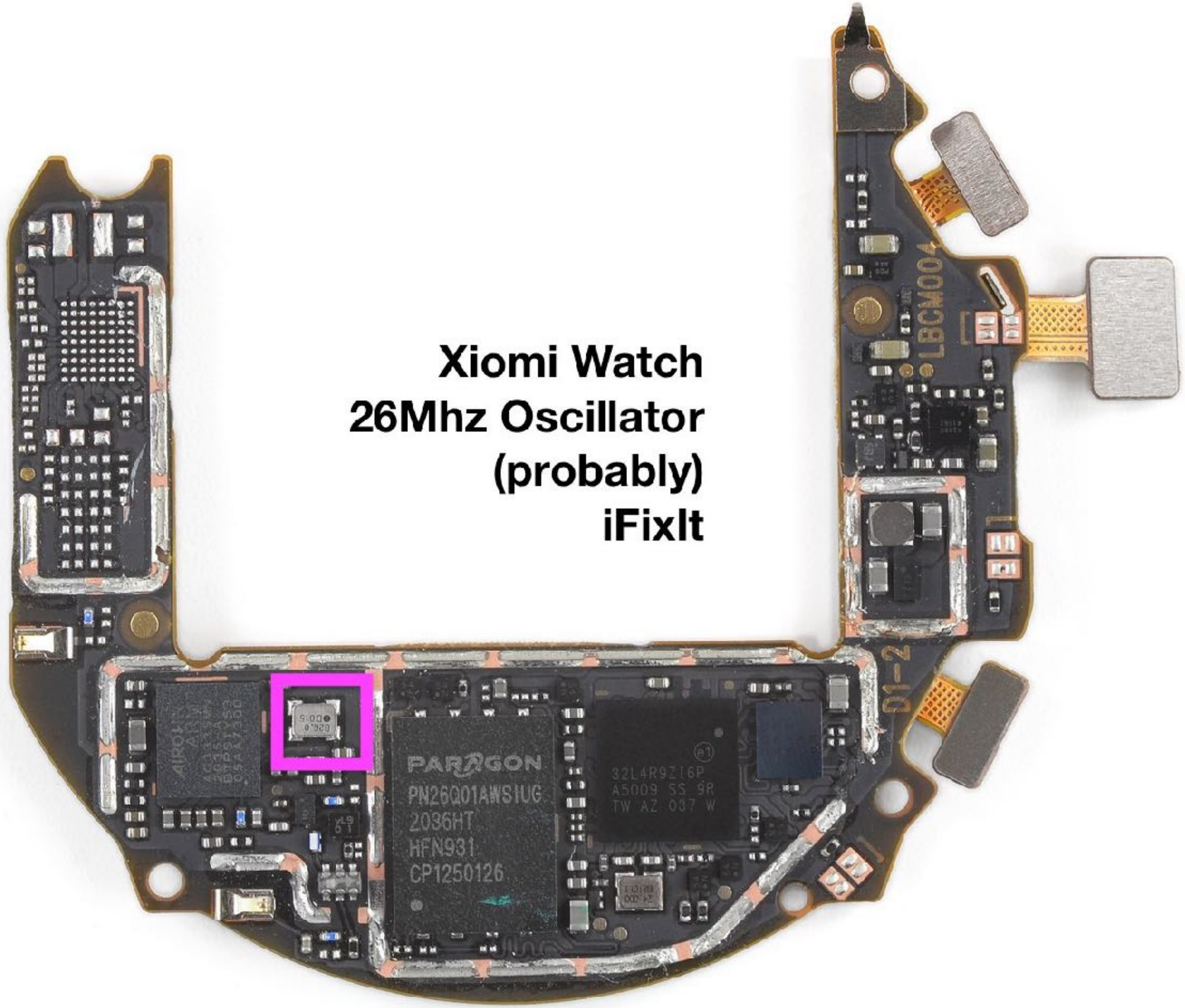
-iFixIt




A10X


24.000

Xiomi Watch
26Mhz Oscillator
(probably)
iFixIt



NBVSBAXXX Series

2.5 V/3.3 V, LVPECL Voltage-Controlled Crystal Oscillator (VCXO) PureEdge™ Product Series

ON Semiconductor®

http://onsemi.com

The NBVSBAXXX series voltage−controlled crystal oscillator (VCXO) devices are designed to meet today's requirements for 2.5 V and 3.3 V LVPECL clock generation applications. These devices use a high Q fundamental mode crystal and Phase Locked Loop (PLL) multiplier to provide a wide range of frequencies from 60 MHz to 700 MHz (factory configurable per user specifications) with a pullable range of ±100 ppm and a frequency stability of ±50 ppm. The silicon−based PureEdge™ products design provides users with exceptional frequency stability and reliability. They produce an ultra low jitter and phase noise LVPECL differential output.

The NBVSBAXXX series are members of ON Semiconductor's PureEdge™ clock family that provides accurate and precision clock generation solutions.

Available in the industry standard 5.0 x 7.0 x 1.8 mm and in a new 3.2 x 5.0 x 1.2 mm SMD (CLCC) package on 16 mm tape and reel in quantities of 1,000.

Features

- LVPECL Differential Output
- Operating Range: 2.5 V ±5%, 3.3 V ±10%

MARKING DIAGRAM

6 PIN CLCC
LN SUFFIX
CASE 848AB

NBVSBAXXX
XXX.XXXX
AWLYYWWG

6 PIN CLCC
LU SUFFIX
CASE 848AC

NBVSBAXXX
XXX.XXXX
AWLYYWWG

NBVSBAXXX = NBVSBAXXX (±50 ppm)
XXX.XXXX = Output Frequency (MHz)
A = Assembly Location

ONSEMI A037L 707.35MHZ VCXO

sketch_sep29a | Arduino 1.8.13
sketch_sep29a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
sketch_200929a | Processing 3.5.3
Java
sketch_200929a
void setup() {
}
void draw() {
}
Console
Errors
Updates
p5.js Web Editor
editor.p5js.org
File
Edit
Sketch
Help
English
Hello, jfeddersen!
Auto-refresh
Nettle cardboard
sketch.js
Preview
function setup() {
createCanvas(400, 400);
}
function draw() {
background(220);
}
Console
Clear
ARDUINO
PROCESSING
P5JS
"TICK"


p5.js/main.js at 30d338b55bb
github.com/processing/p5.js/blo
360 };
362 this._draw =
363 const now =
364 const time_
365 const targe
367 // only dra
368 // draw if
369 // (this wi
370 // again whe
371 // issue whe
372 // in sync
373 // if loopi
374 // is the c
375 const epsil
376 if (
377 !this._lo
378 time_sinc
380 //mandato
381 this.redraw();
p5.js Web Editor | TimeoutInter
editor.p5js.org/jfeddersen/sketches/vqW-vpWJR
Update
File
Edit
Sketch
Help
Paused in debugger
English
Hello, jfeddersen!
Auto-refresh
TimeoutIntervalBasic
by jfeddersen
sketch.js
Saved: about 1 year ago
Preview
Console
Clear
Hi from Interval: 1148.699999988079
Hi from Interval: 623279.5
***Yo from Timeout: 623279.6000000238
Stopping intervalID 2
Elements
Console
Sources
Network
Performance
12
7
Page
p5.sound.min.js:formatted
top
editor.p5js.org
jfeddersen/s
vqW-vpW
app.f3cea0a
app.f3cea0a
ff35e397426
fonts.googleap
fonts.gstatic.c
webpack://
www.google-a
preview.p5js.o
186f6e77-68da-4
6a19a454-faba-4
t.Context.prototype.now = function() {
return this._context.currentTime
}
t.Context.prototype._createWorker = funct
window.URL = window.URL || window.wel
var t = new Blob(["var timeoutTime =
, e = URL.createObjectURL(t)
, i = new Worker(e);
return i.addEventListener("message",
this.emit("tick")
}
.bind(this)),
i.addEventListener("message", functio
var t = this.now();
if (this.isNumber(this._lastUpdat
var e = t - this._lastUpdate;
this._computedUpdateInterval
}
2 lines, 34 characters selected
Coverage: n/a
Debugger paused
Threads
Main
paused
186f6e77-68da-4004-a763-fe2…
6a19a454-faba-4e3e-a750-08…
Watch
Breakpoints
vqW-vpWJR:1
Scope
Local
this: t.Context
Closure
Closure
Script
ardboard
Preview
JS
Clear
ick"

EASING

Smoothly transition a variable from one value to another in a set time

SIMULATION

Use physics or other rules to determine next frame for one or more objects.

TIMELINES

Schedule code for execution in the future

EASING

ROBERT PENNER

Quadratic Easing

Flash's Timeline tweens use something called *quadratic easing*—which could actually be termed "normal" easing. The word *quadratic* refers to the fact that the equation for this motion is based on a squared variable, in this case, t^2:

$$p(t) = t^2$$

NOTE: I always wondered why the term *quadratic* (the prefix means "four") is used to describe equations with a degree of two (x^2). While writing this chapter, I finally looked it up in the dictionary (RTFD, you might say). I discovered that *quad* originally referred to the four sides of a square. Thus, a *squared* variable is *quadratic*.

I used the quadratic easing curve earlier in Figure 7-4. It's actually half a parabola. Here it is again, for reference purposes, in Figure 7-7.

Here's the quadratic ease-in ActionScript function:

```
Math.easeInQuad = function (t, b, c, d) {
    return c*(t/=d)*t + b;
};
```

Recall that `t` is time, `b` is beginning position, `c` is the total change in position, and `d` is the duration of the tween.

This equation is more complex than the linear tween, but it's the simplest of the equations that implement easing. Basically, I normalize `t` by dividing it by `d`. This forces `t` to fall between 0 and 1. I multiply `t` by itself to produce quadratic curvature in the values. Then I scale the value from a

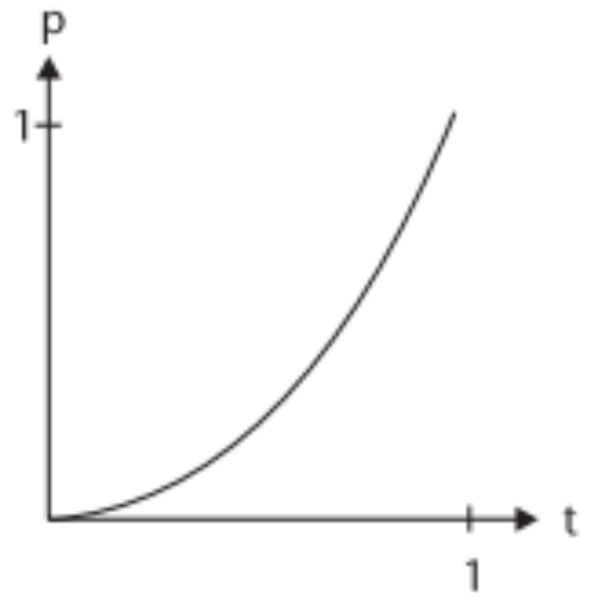
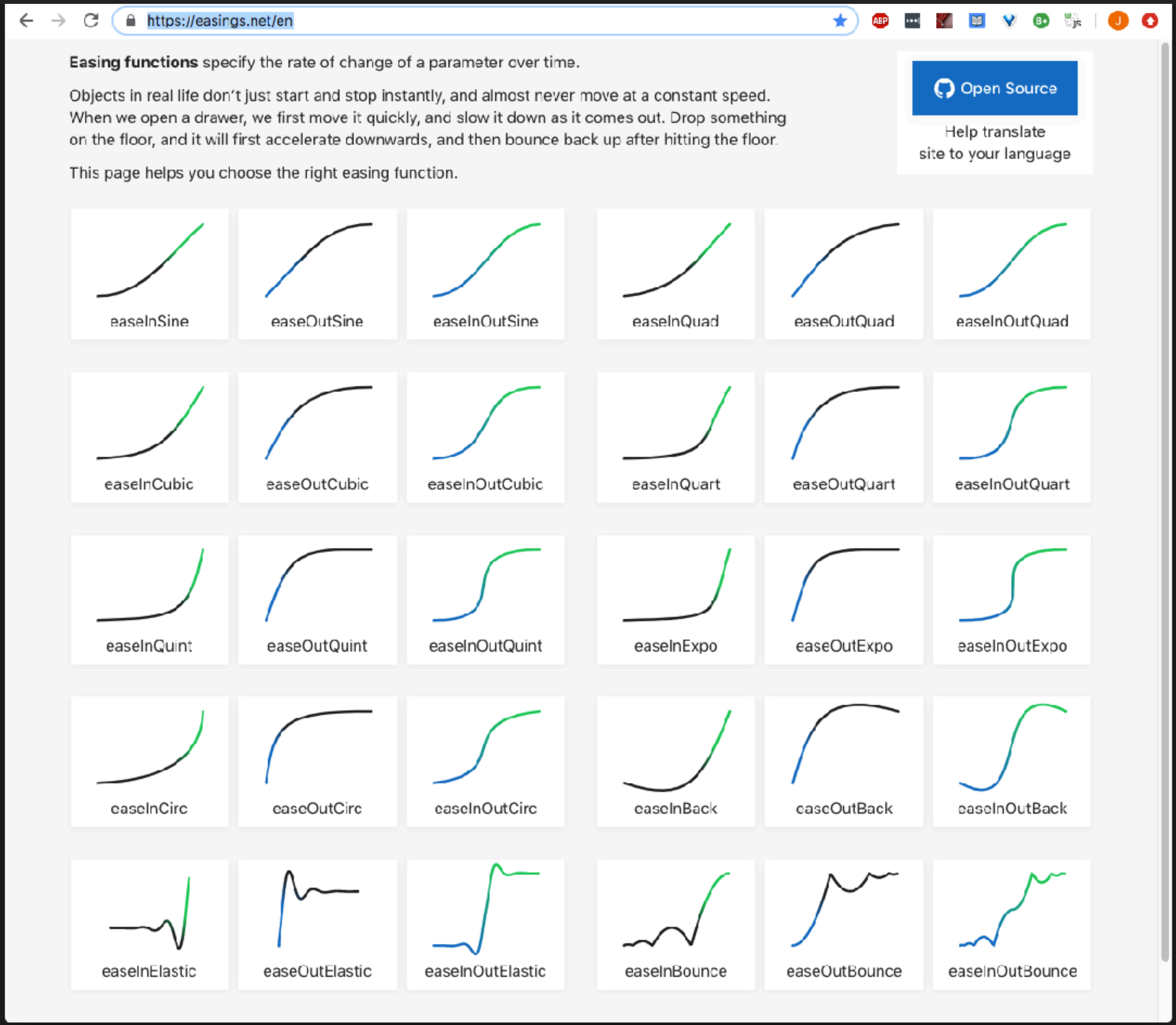


FIGURE 7-7

Graph of quadratic easing

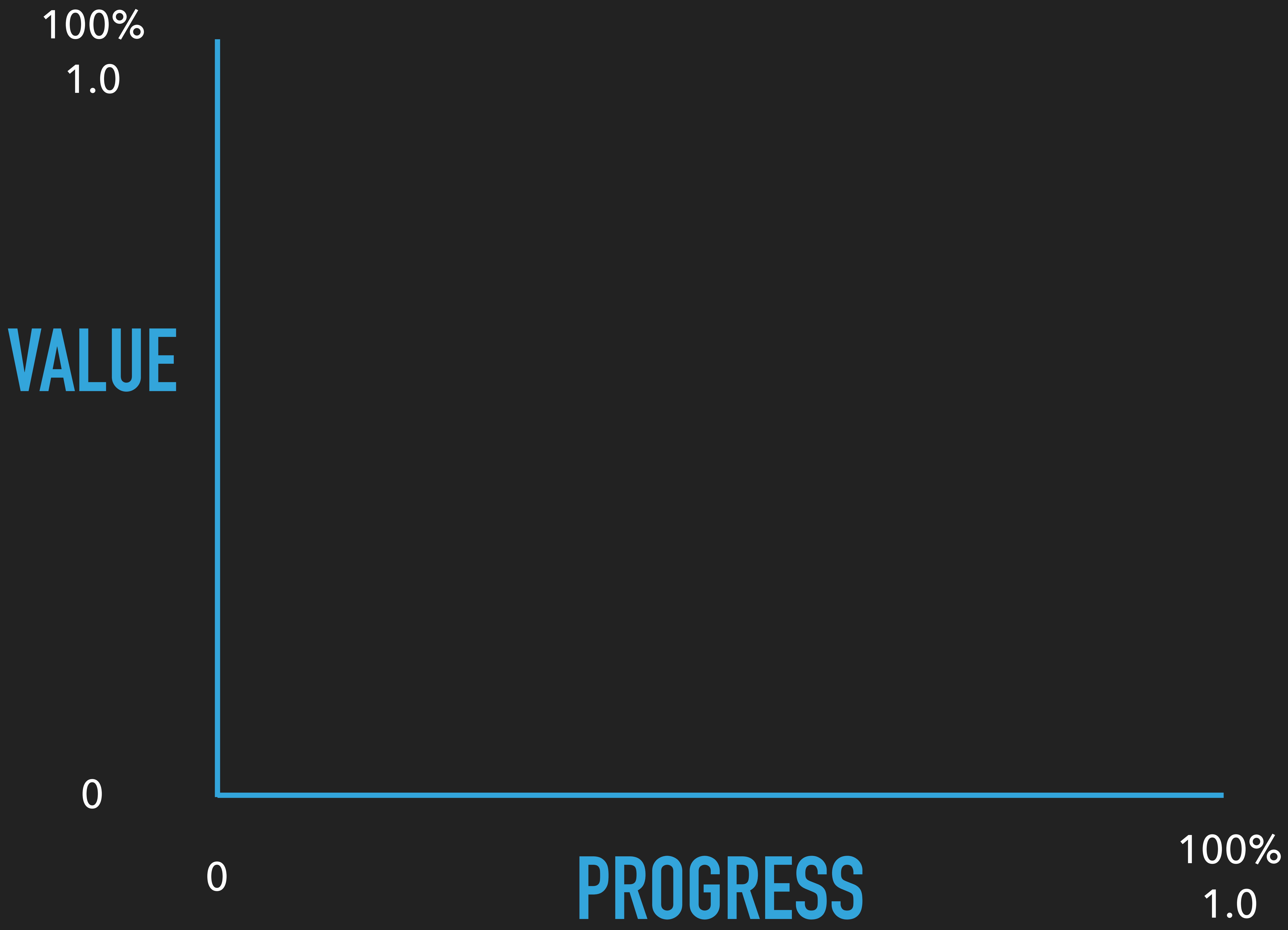
Robert Penner's Programming Macromedia Flash (2002)

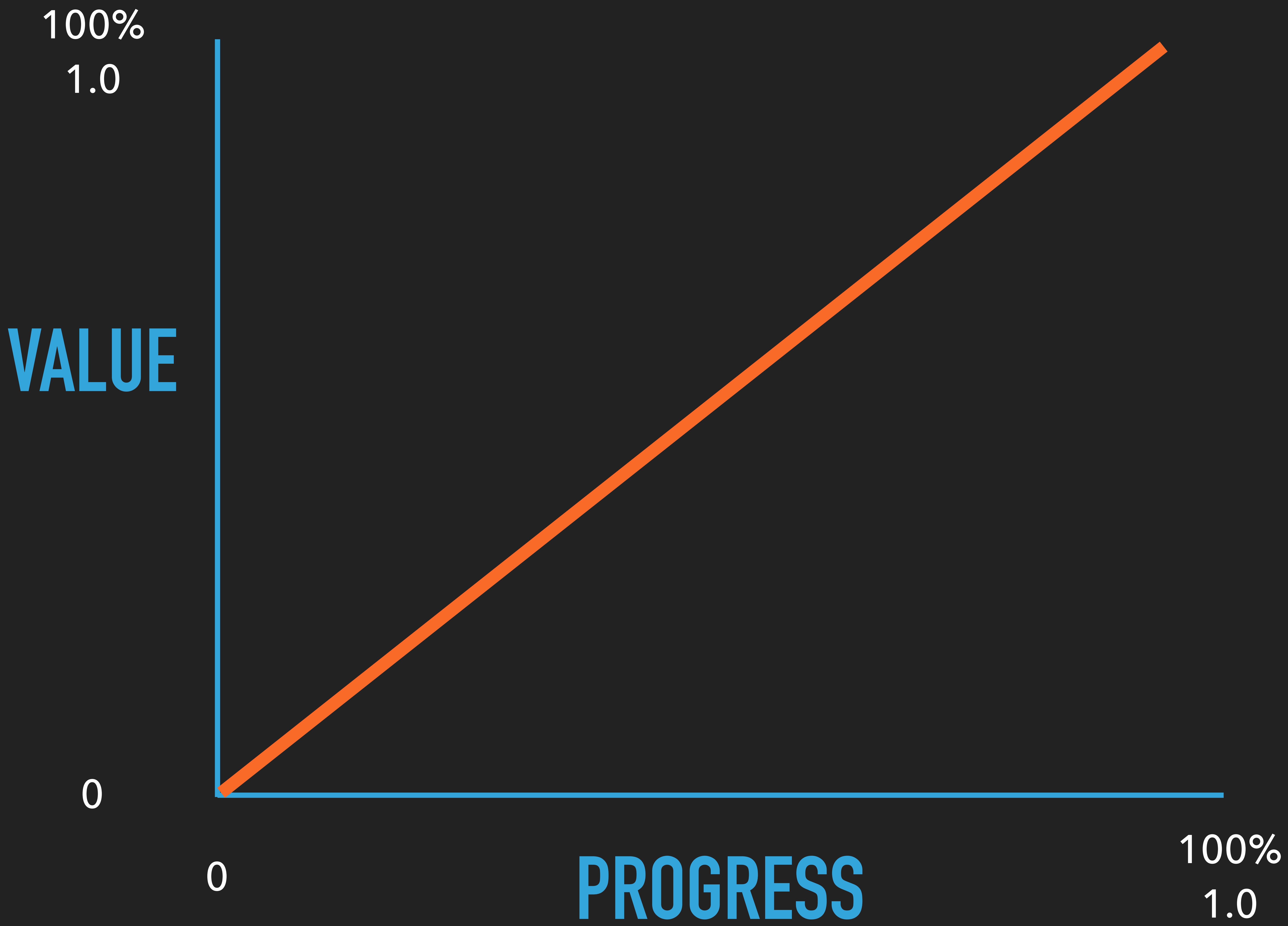
ROBERT PENNER



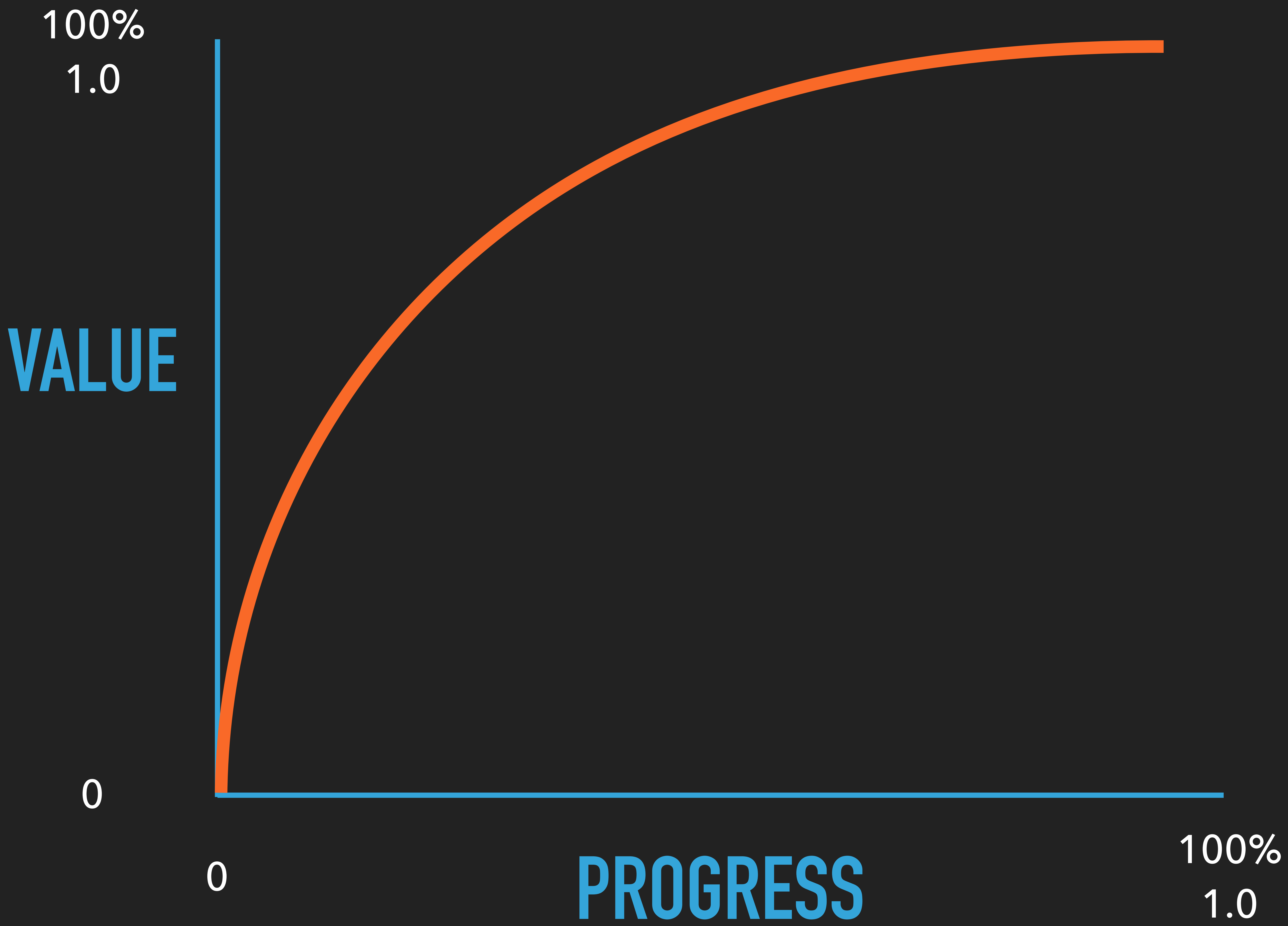
https://easings.net/en

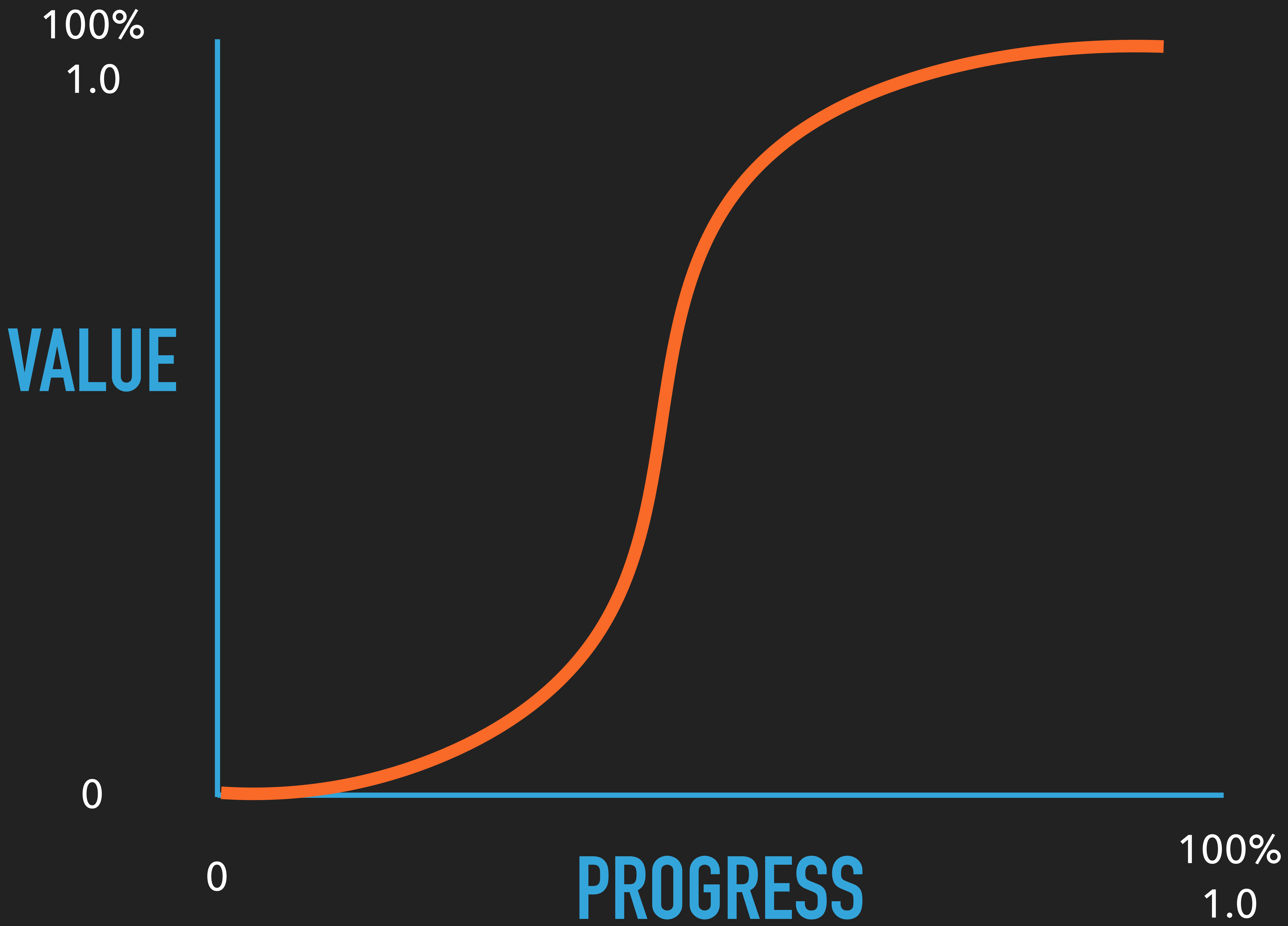
EASING Smoothly transition a variable from one value to another in a set time

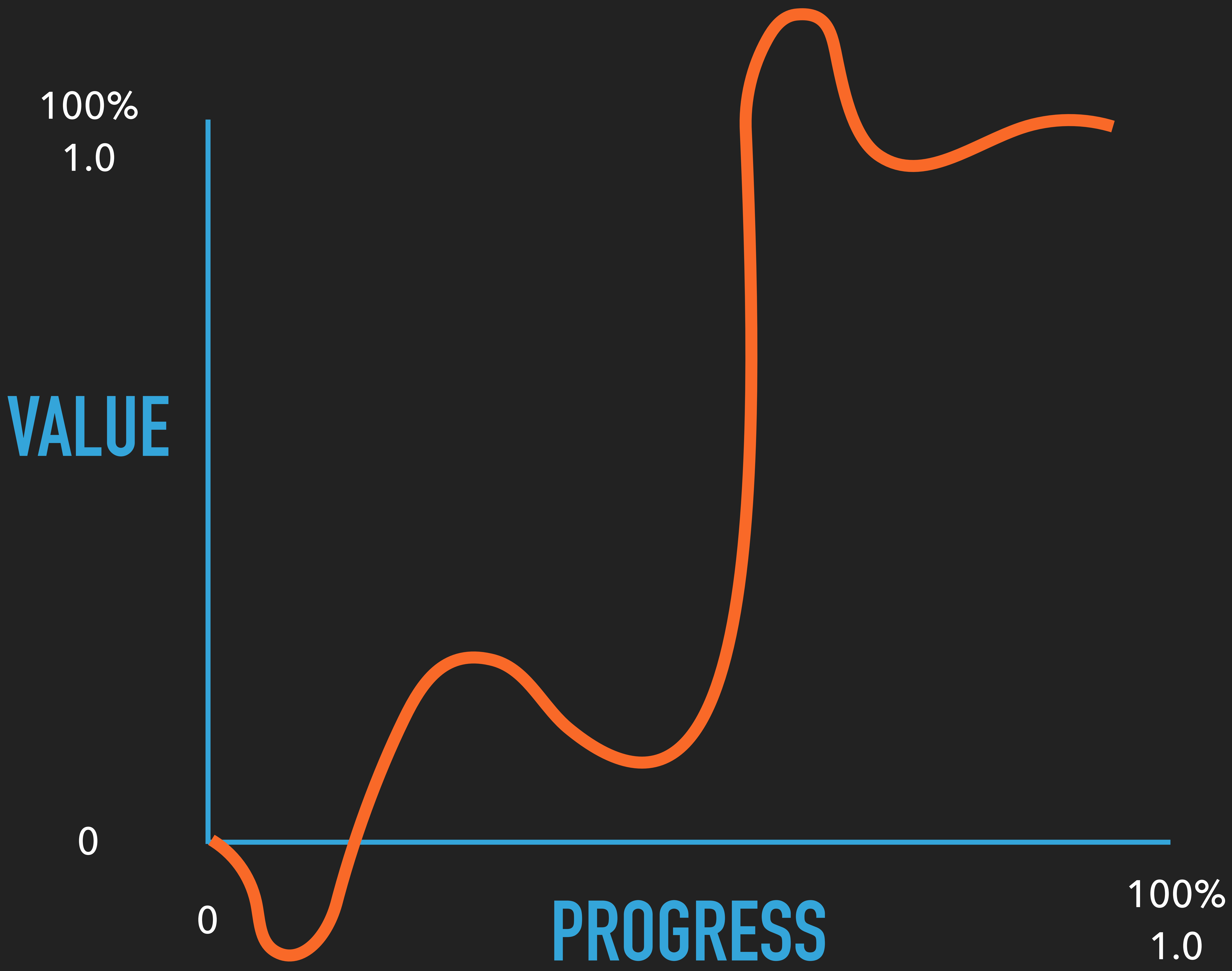




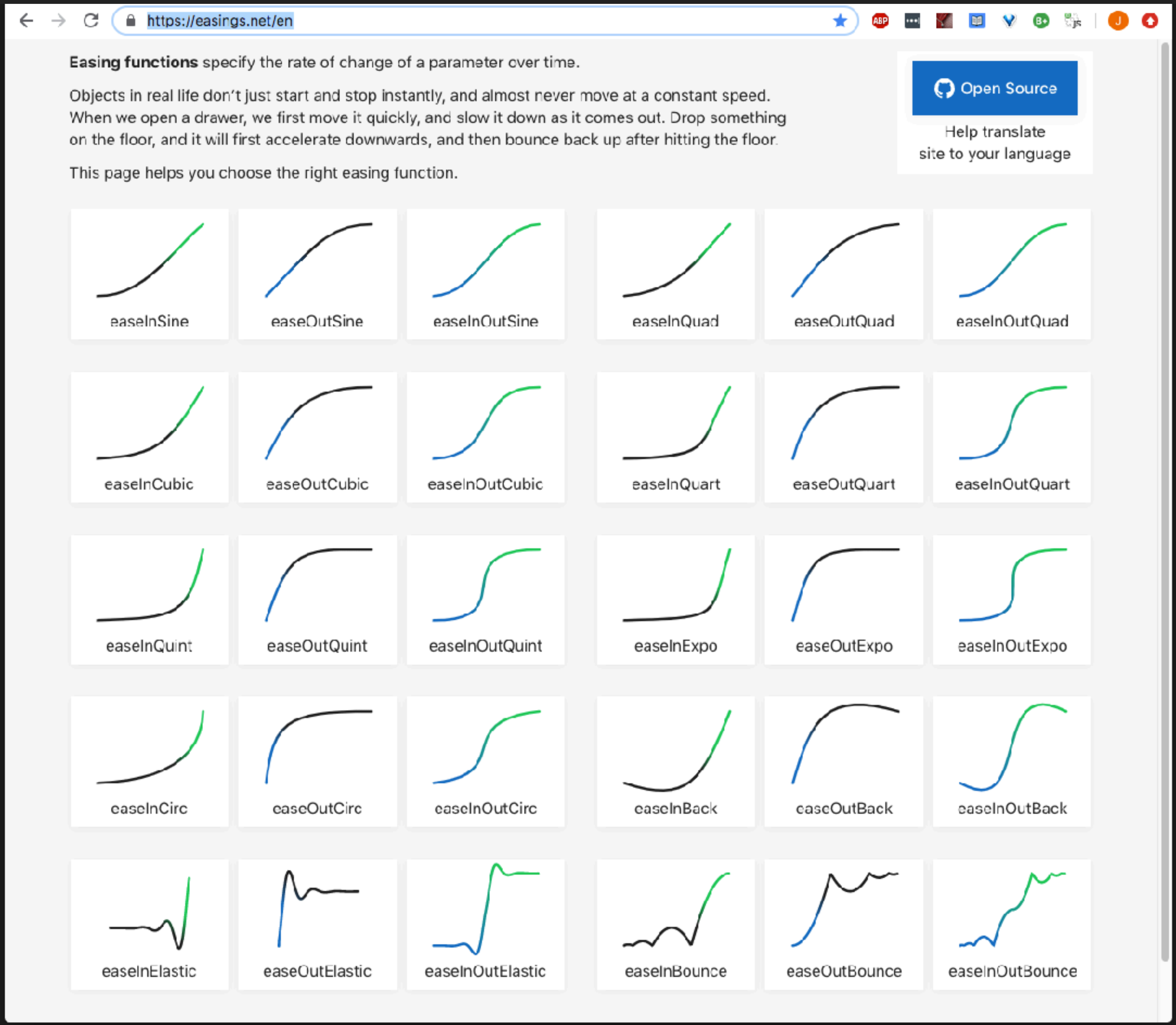
100%
1.0
VALUE
0
0
PROGRESS
100%
1.0


100%
1.0
VALUE
0
0
PROGRESS
100%
1.0


100%
1.0
VALUE
0
0
PROGRESS
100%
1.0


100%
1.0
VALUE
0
0
PROGRESS
100%
1.0

ROBERT PENNER



https://easings.net/en

ROBERT PENNER

Quadratic Easing

Flash's Timeline tweens use something called *quadratic easing*—which could actually be termed "normal" easing. The word *quadratic* refers to the fact that the equation for this motion is based on a squared variable, in this case, t^2:

$$p(t) = t^2$$

NOTE: I always wondered why the term *quadratic* (the prefix means "four") is used to describe equations with a degree of two (x^2). While writing this chapter, I finally looked it up in the dictionary (RTFD, you might say). I discovered that *quad* originally referred to the four sides of a square. Thus, a *squared* variable is *quadratic*.

I used the quadratic easing curve earlier in Figure 7-4. It's actually half a parabola. Here it is again, for reference purposes, in Figure 7-7.

Here's the quadratic ease-in ActionScript function:

```
Math.easeInQuad = function (t, b, c, d) {
    return c*(t/=d)*t + b;
};
```

Recall that `t` is time, `b` is beginning position, `c` is the total change in position, and `d` is the duration of the tween.

This equation is more complex than the linear tween, but it's the simplest of the equations that implement easing. Basically, I normalize `t` by dividing it by `d`. This forces `t` to fall between 0 and 1. I multiply `t` by itself to produce quadratic curvature in the values. Then I scale the value from a

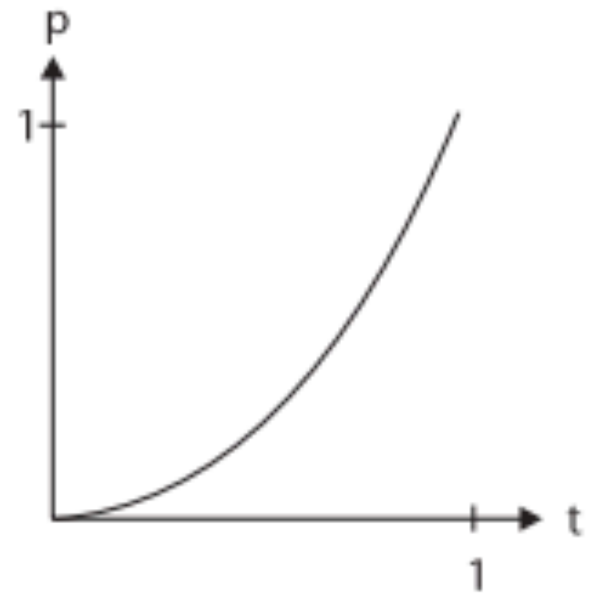
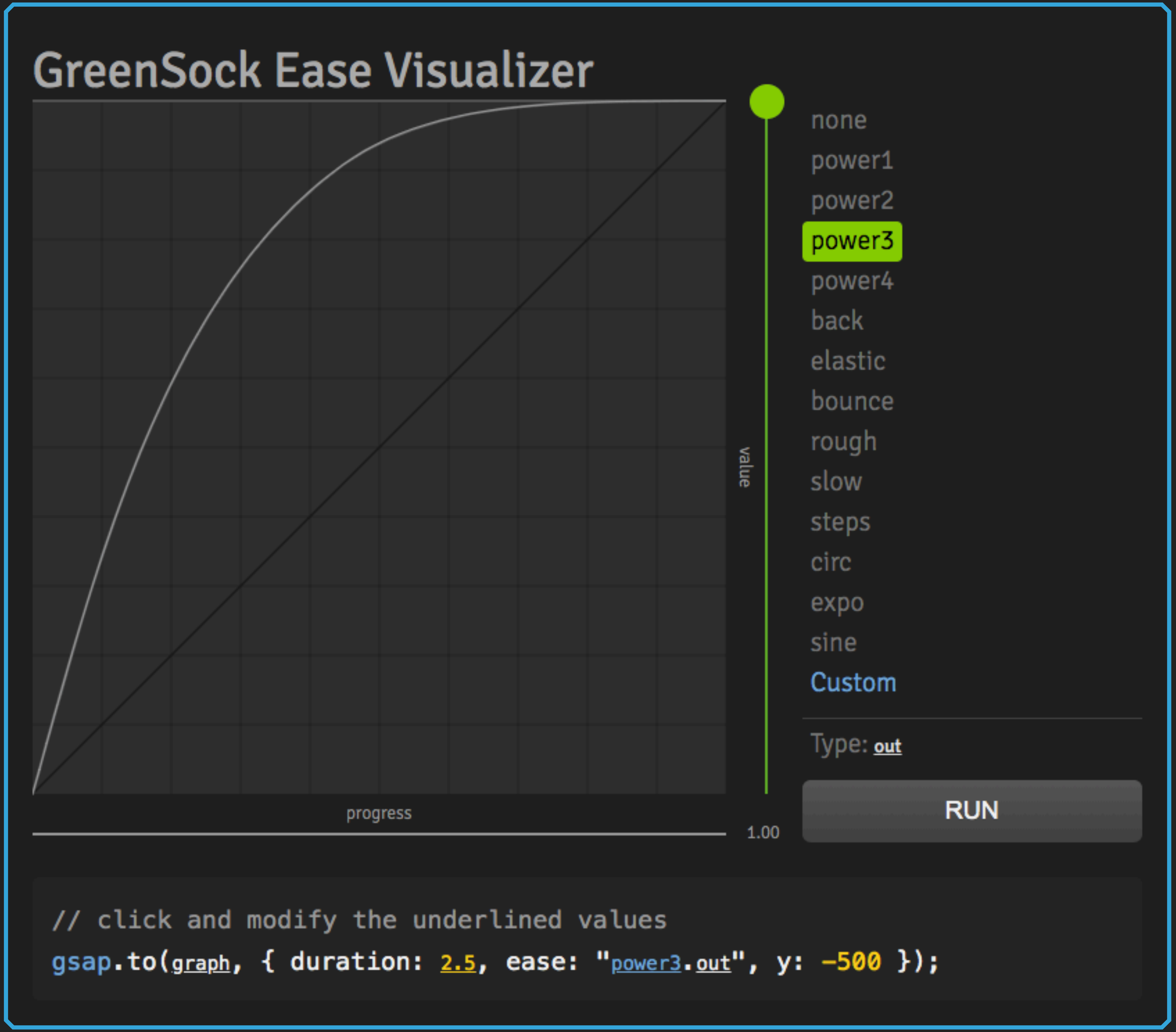


FIGURE 7-7

Graph of quadratic easing

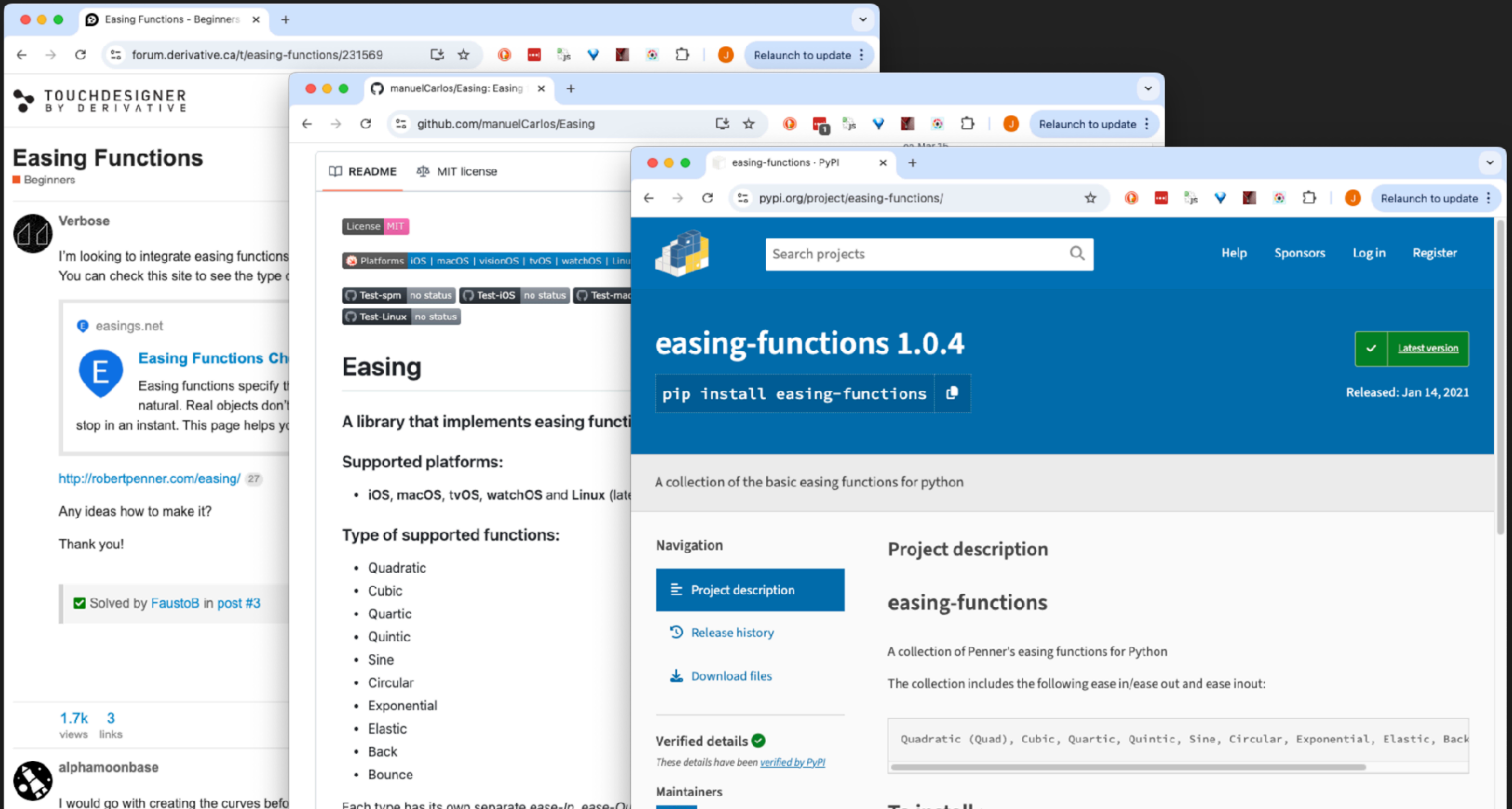
Robert Penner's Programming Macromedia Flash (2002)



https://greensock.com/ease-visualizer

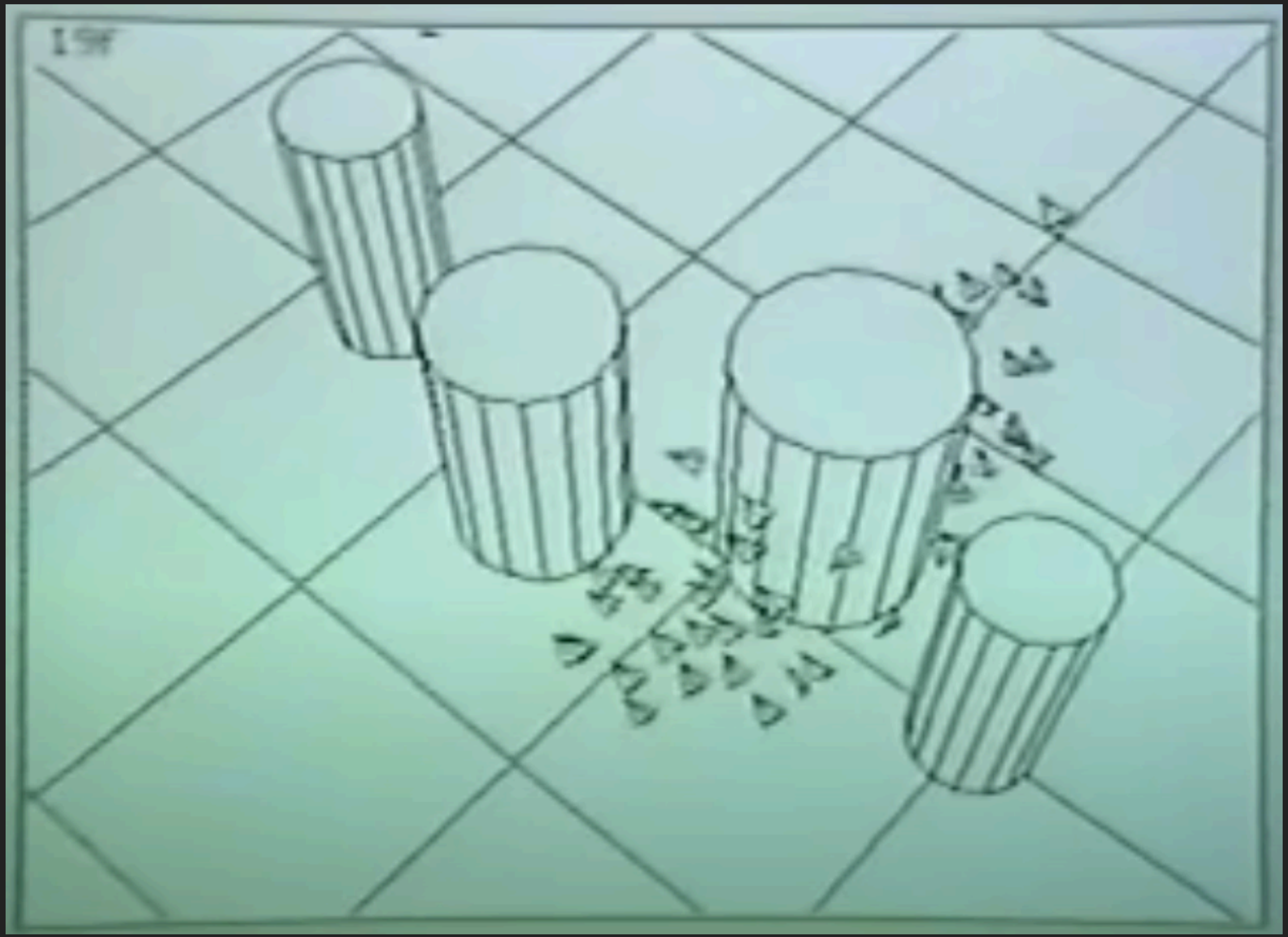
Fine Tune Keyframes
Effect Controls
Effects
Video Effects
Motion
Opacity
Time Remapping
Brightness & Contrast
Brightness
Contrast
Black & White
Velocity: 22,8 / second
00:00:41:20
Project: generiek2
generiek2.prproj
Media Browser
Favorites
Local Drives
Network Drives
MVI_9405.MOV
MVI_9406.MOV
MVI_9411.MOV
MVI_9413.MOV
MVI_9419.MOV
Synth Lead
Snap:
Smart
Drag:
No Overlap
Edit
Options
View
Single
Tracks
All
Audio
Inst
Aux
Bus
Input
Master
MIDI
Bars|Beats
Tempo
Chords
Markers

Once you see “easings”, they’re everywhere!

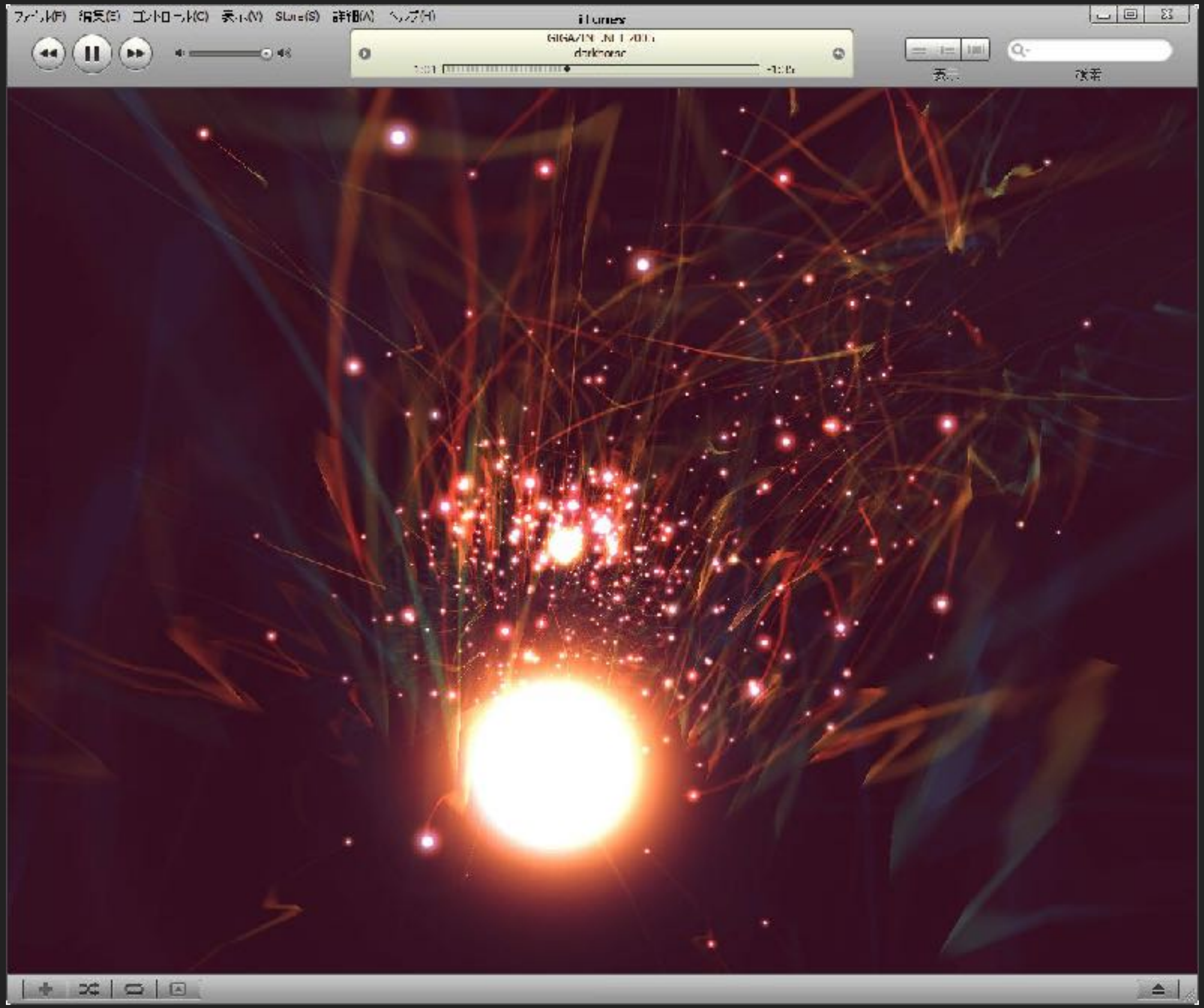


SIMULATION

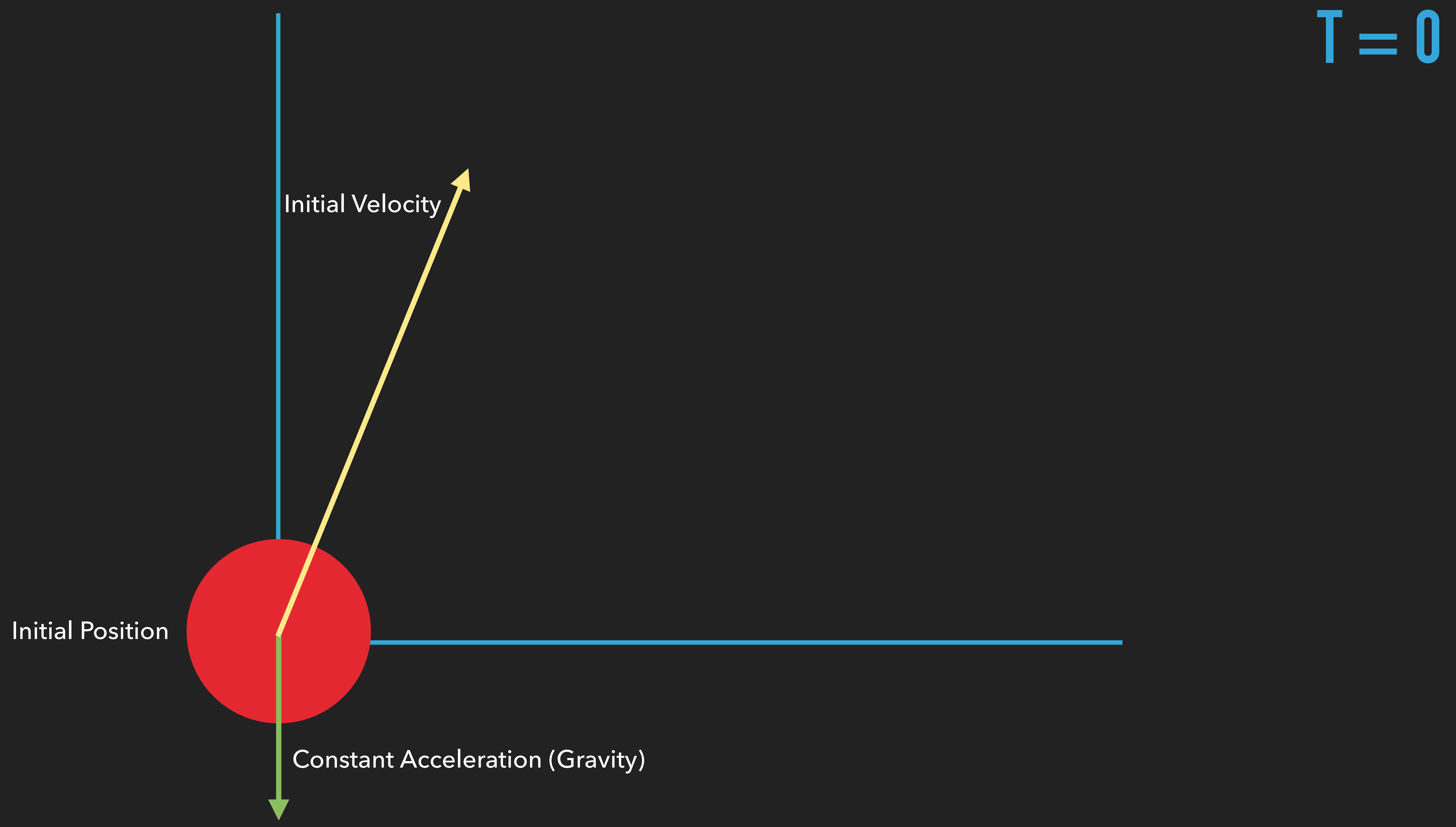
SIMULATION Use physics or other rules to determine next frame for one or more objects.



Craig Reynolds' Boids (1986)

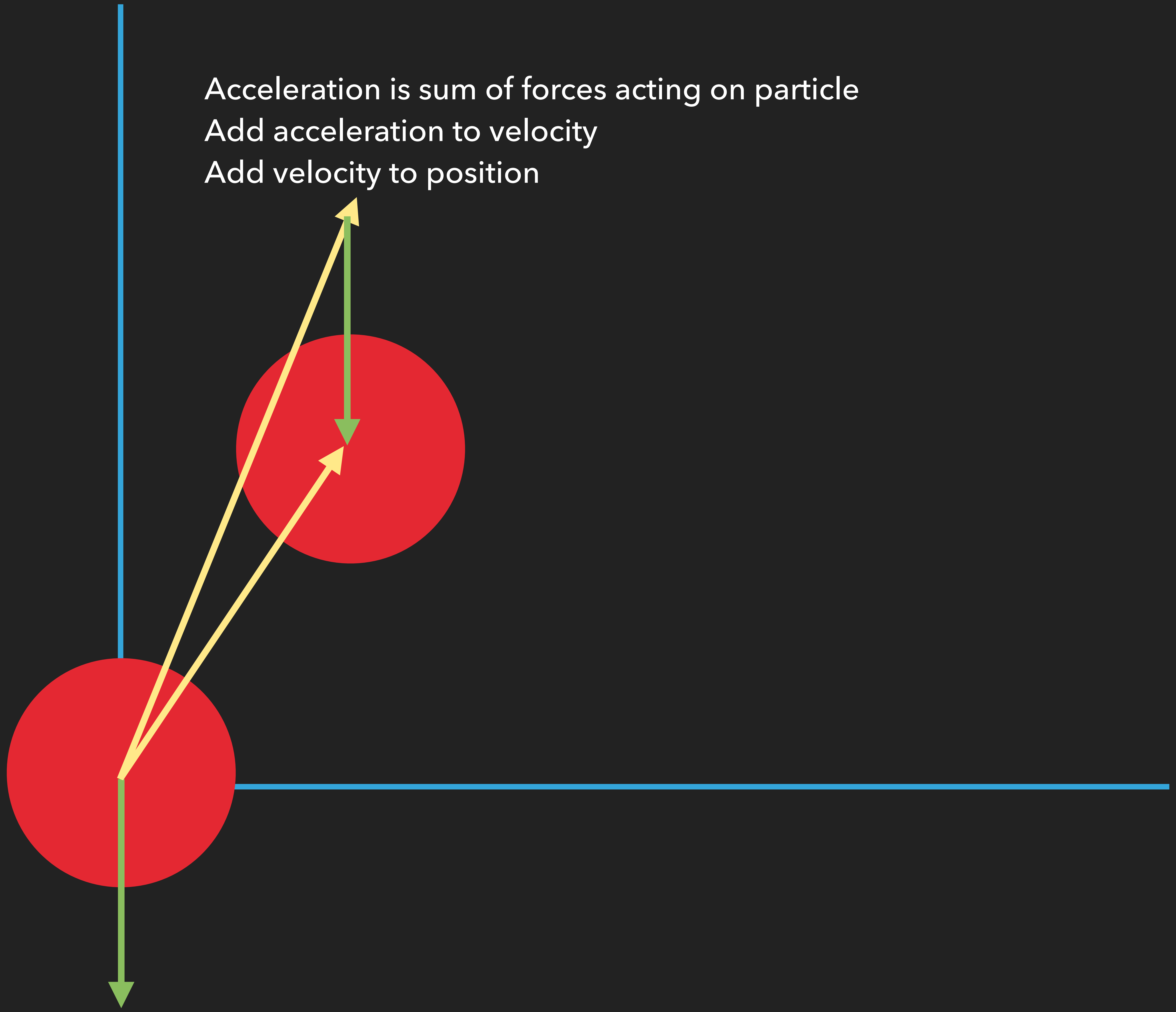


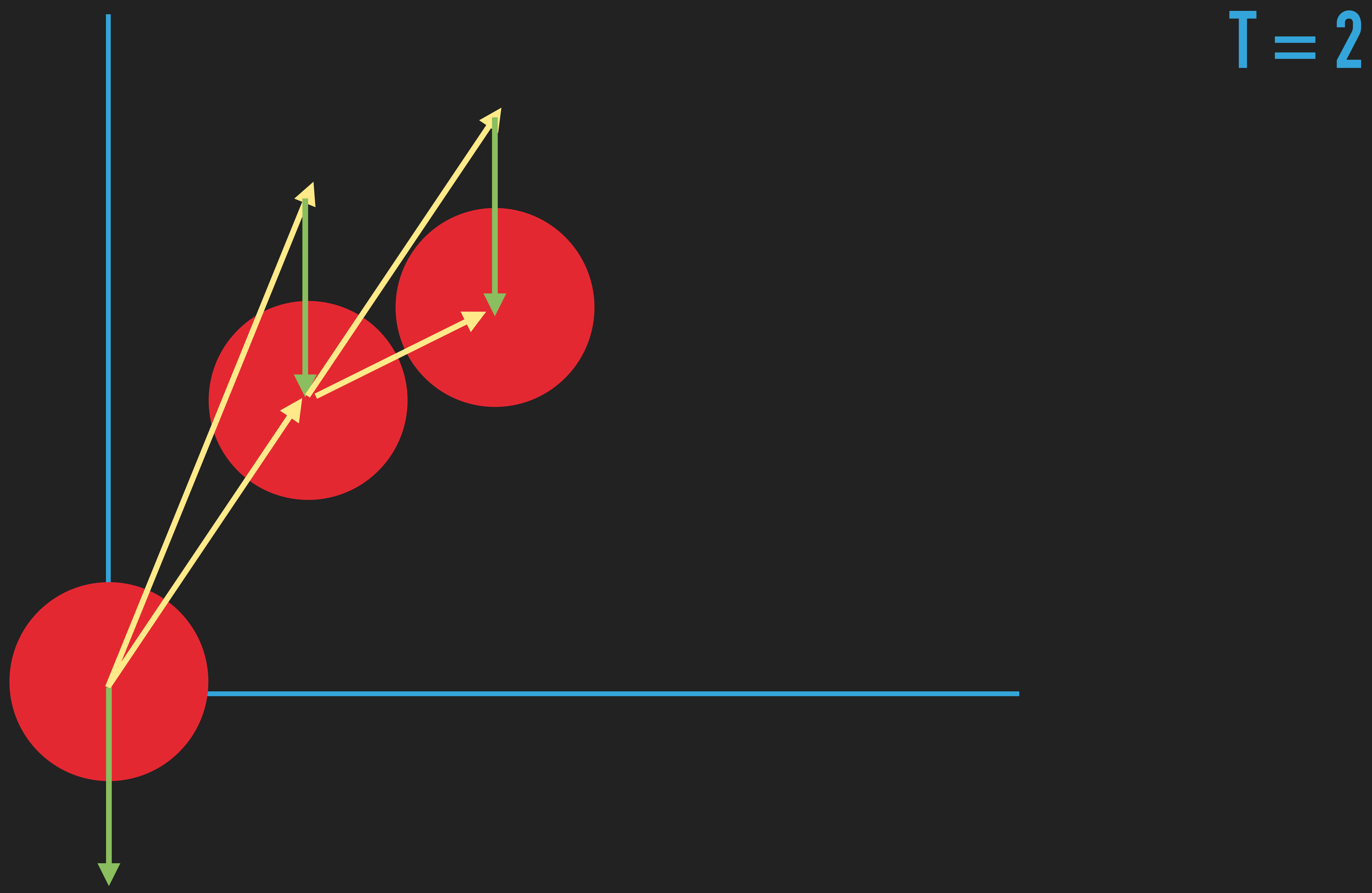
Robert Hodgin's (Flight 404) Magnetosphere, 2007

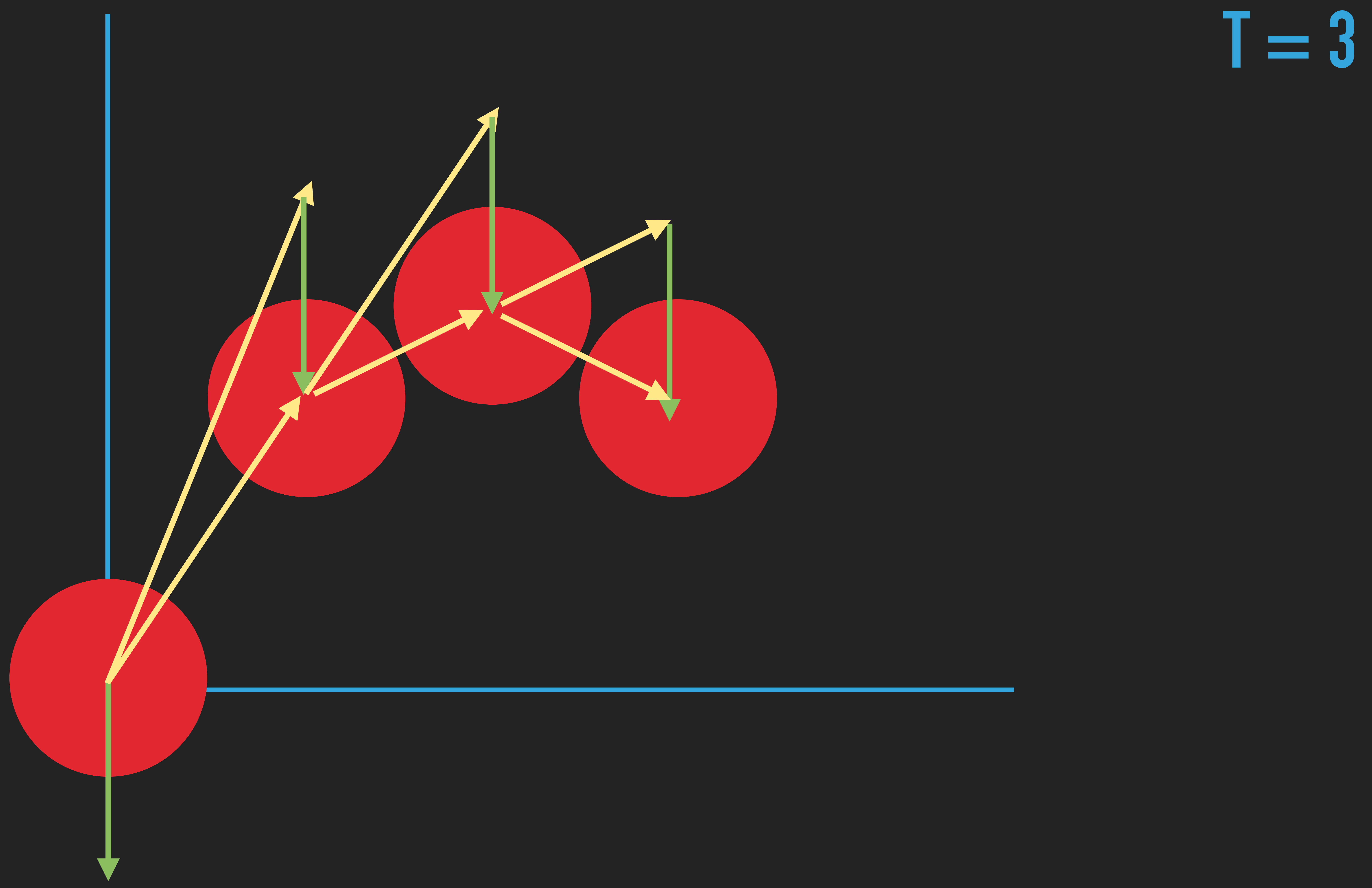

T = 0
Initial Velocity
Initial Position
Constant Acceleration (Gravity)

T = 1

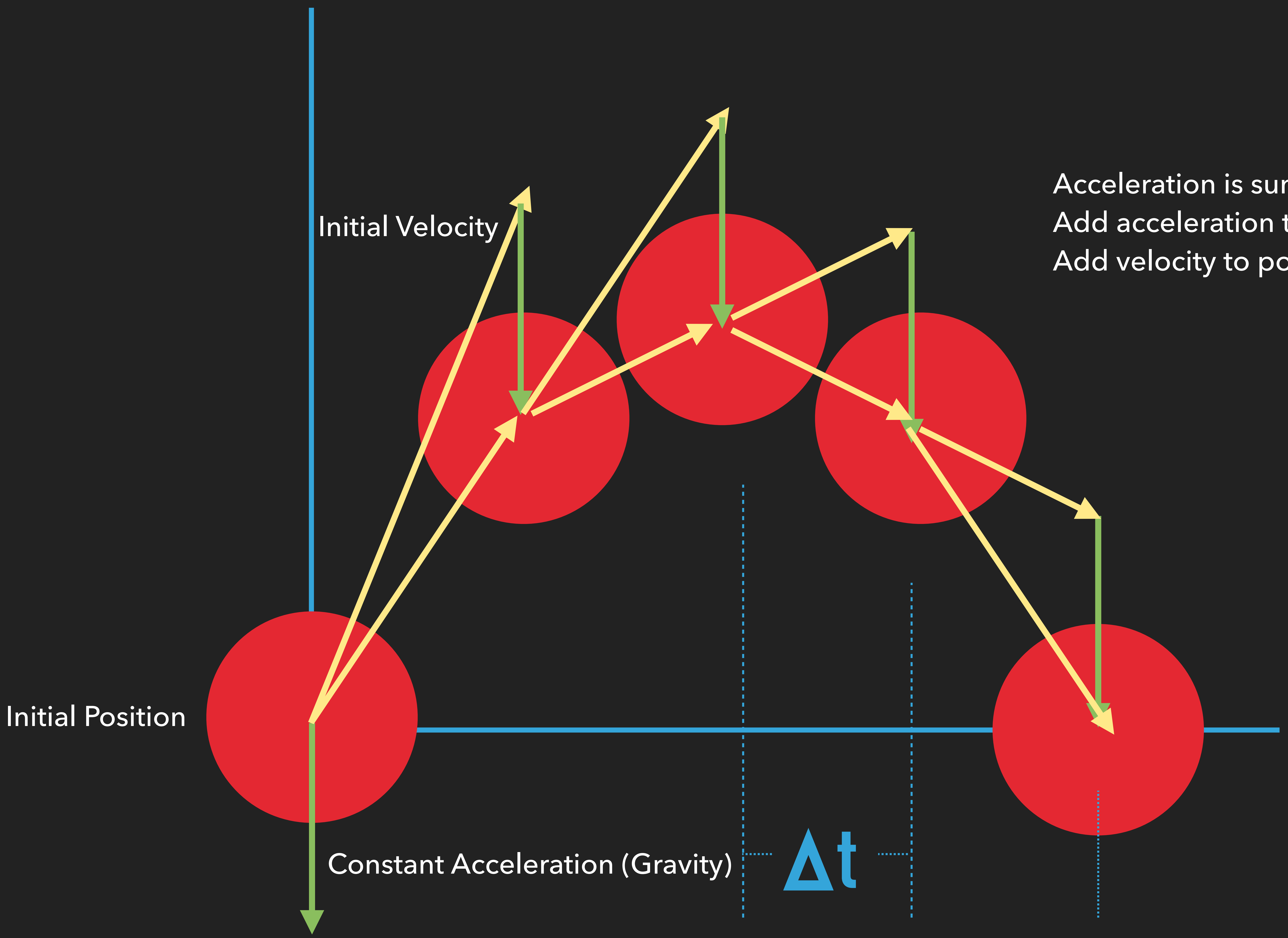
Acceleration is sum of forces acting on particle
Add acceleration to velocity
Add velocity to position




T = 2


T = 3

T = 4...



Acceleration is sum of forces acting on particle
Add acceleration to velocity
Add velocity to position

Can be expanded to three dimensions, multiple particles, and attractive and repulsive forces. But the steps between frames will remain basic vector addition.

http://roberthodgin.com/project/magnetosphere

```
//  Created by Robert Hodgin on 5/14/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#include "cinder/app/AppBasic.h"
#include "cinder/Rand.h"
#include "cinder/Sphere.h"
#include "Particle.h"

using namespace ci;

Particle::Particle(){}

Particle::Particle( const Vec3f &pos, float charge )
        : mPos( pos ), mCharge( charge )
{
        mVel                    = Vec3f::zero();
        mAcc                    = Vec3f::zero();
        mForce                  = 0.0f;

        mRadius                 = 1.0f;
        mShellRadius    = 12.0f;
}

void Particle::update( const Camera &cam, float dt )
{
        Sphere s                = Sphere( mPos, mRadius * 10.0f );
        mScreenPos              = cam.worldToScreen( mPos, app::getWindowWidth(), app::getWindowHeight() );
        mScreenRadius   = cam.getScreenRadius( s, app::getWindowWidth(), app::get

        mColor                  = mCharge * 0.5f + 0.5f;

        mVel += mAcc * dt;
        mPos += mVel * dt;
        mAcc = Vec3f::zero();

        mShellRadius = mRadius + fabs( mForce ) * 50000.0f;

        mMatrix.setToIdentity();
        mMatrix.translate( mPos );
}

void Particle::draw()
{
        gl::color( Color( mColor, mColor, mColor ) );
        gl::drawSphere( mPos, mRadius );
```


```
mVel += mAcc * dt;
mPos += mVel * dt;
mAcc = Vec3f::zero();
```


Acceleration is sum of forces acting on particle
Add acceleration to velocity
Add velocity to position

https://github.com/flight404/Eyeo2012

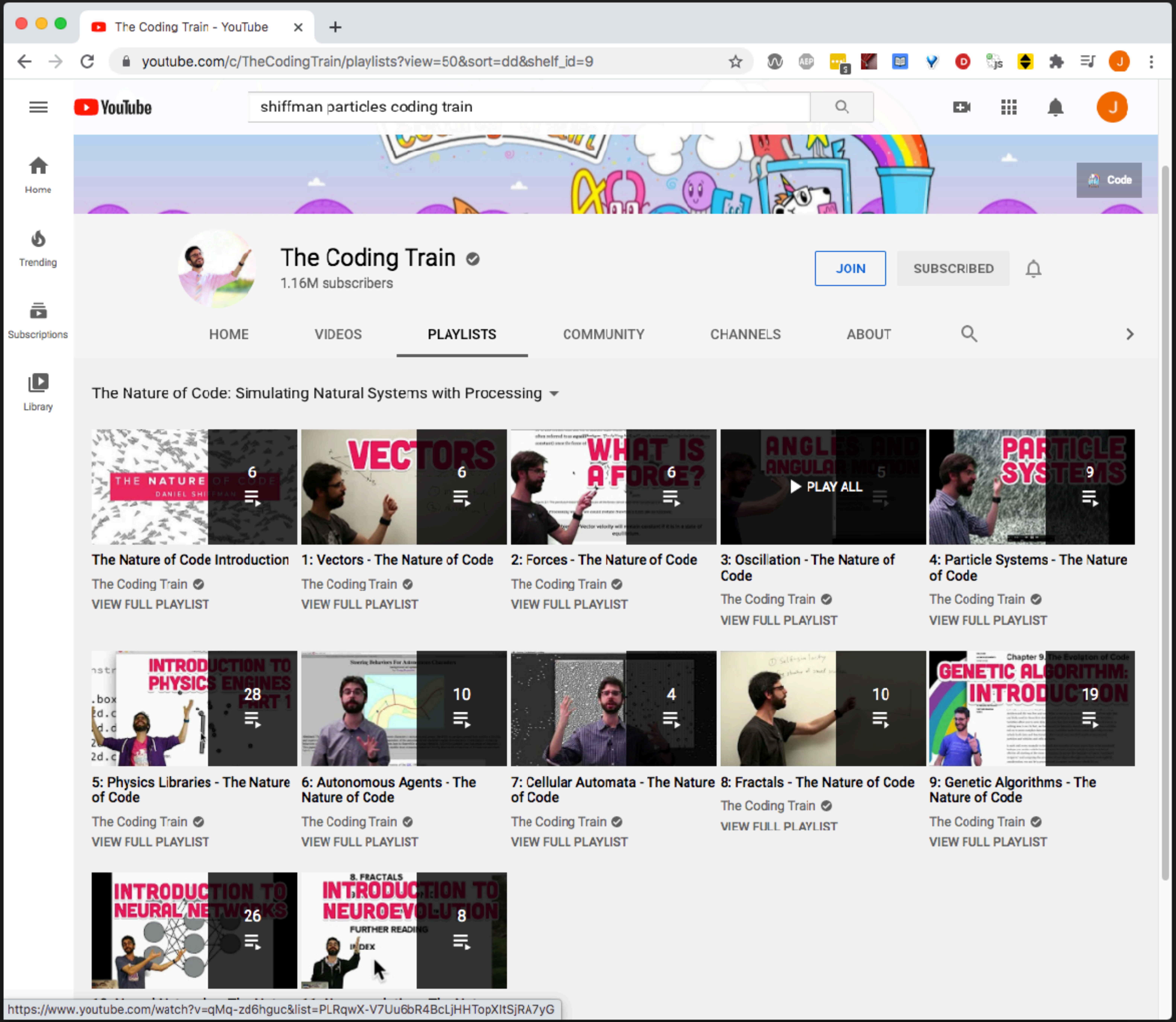
"The physics of the simple vehicle model is based on forward Euler integration. **At each simulation step, behaviorally determined steering forces (as limited by *max_force*) are applied to the vehicle's point mass.** This produces an acceleration equal to the steering force divided by the vehicle's mass. That acceleration is added to the old velocity to produce a new velocity, which is then truncated by *max_speed*. Finally, the velocity is added to the old position:

```
    steering_force = truncate (steering_direction,
max_force)
    acceleration = steering_force / mass
    velocity = truncate (velocity + acceleration,
max_speed)
    position = position + velocity
```

Acceleration is sum of forces acting on particle
Add acceleration to velocity
Add velocity to position

The simple vehicle model maintains its velocity-aligned local space by *incremental adjustment* from the previous time step."

Steering Behaviors For Autonomous Characters
Craig W. Reynolds
https://www.red3d.com/cwr/steer/gdc99/

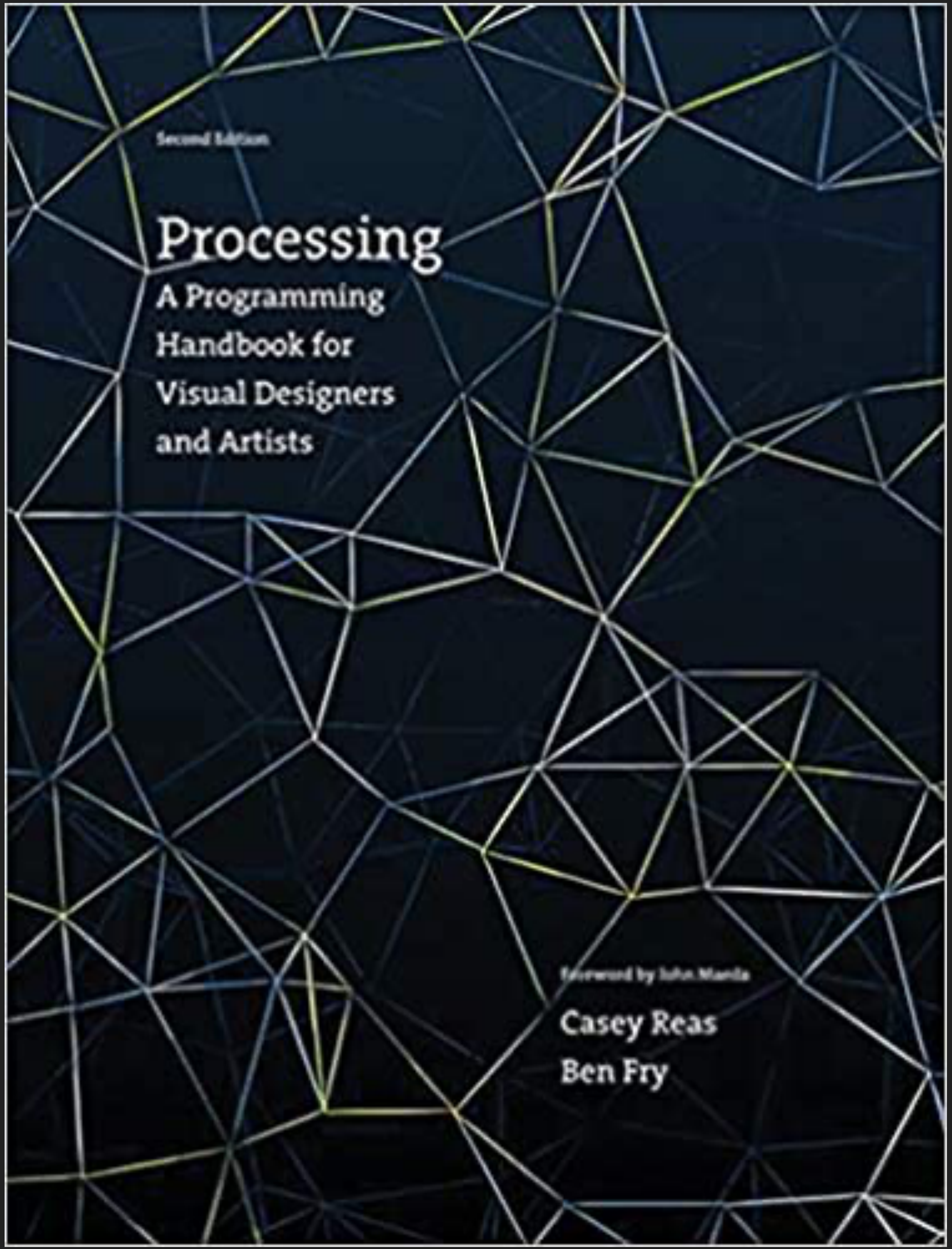


Pretty good source in-house

EXAMPLE

Integrator class used throughout Ben Fry's first Processing text

A simple, simulation-based easing that can easily be applied to variables to achieve smooth animation.



EXAMPLE

Integrator class used throughout Ben Fry's first Processing text

```
class Integrator {

  final float DAMPING = 0.5f;
  final float ATTRACTION = 0.2f;

  float value; float vel; float accel; float force;
  float mass = 1;

  float damping = DAMPING;
  float attraction = ATTRACTION;
  boolean targeting;
  float target;

  Integrator() { }

  Integrator(float value) {
    this.value = value;
  }

  Integrator(float value, float damping,
             float attraction) {
    this.value = value;
    this.damping = damping;
    this.attraction = attraction;
  }

  void set(float v) { value = v;}

  void update() {
    if (targeting) {
      force += attraction * (target - value);
    }

    accel = force / mass;
    vel = (vel + accel) * damping;
    value += vel;
    force = 0;
  }

  void target(float t) {
    targeting = true;
    target = t;
  }

  void noTarget() {
    targeting = false;
  }
}
```

What this means: An **Integrator** maintains a **current value**. That value can be set to a new **target**. With each call to **update**, it will use a simple simulation to move the current value towards the target.

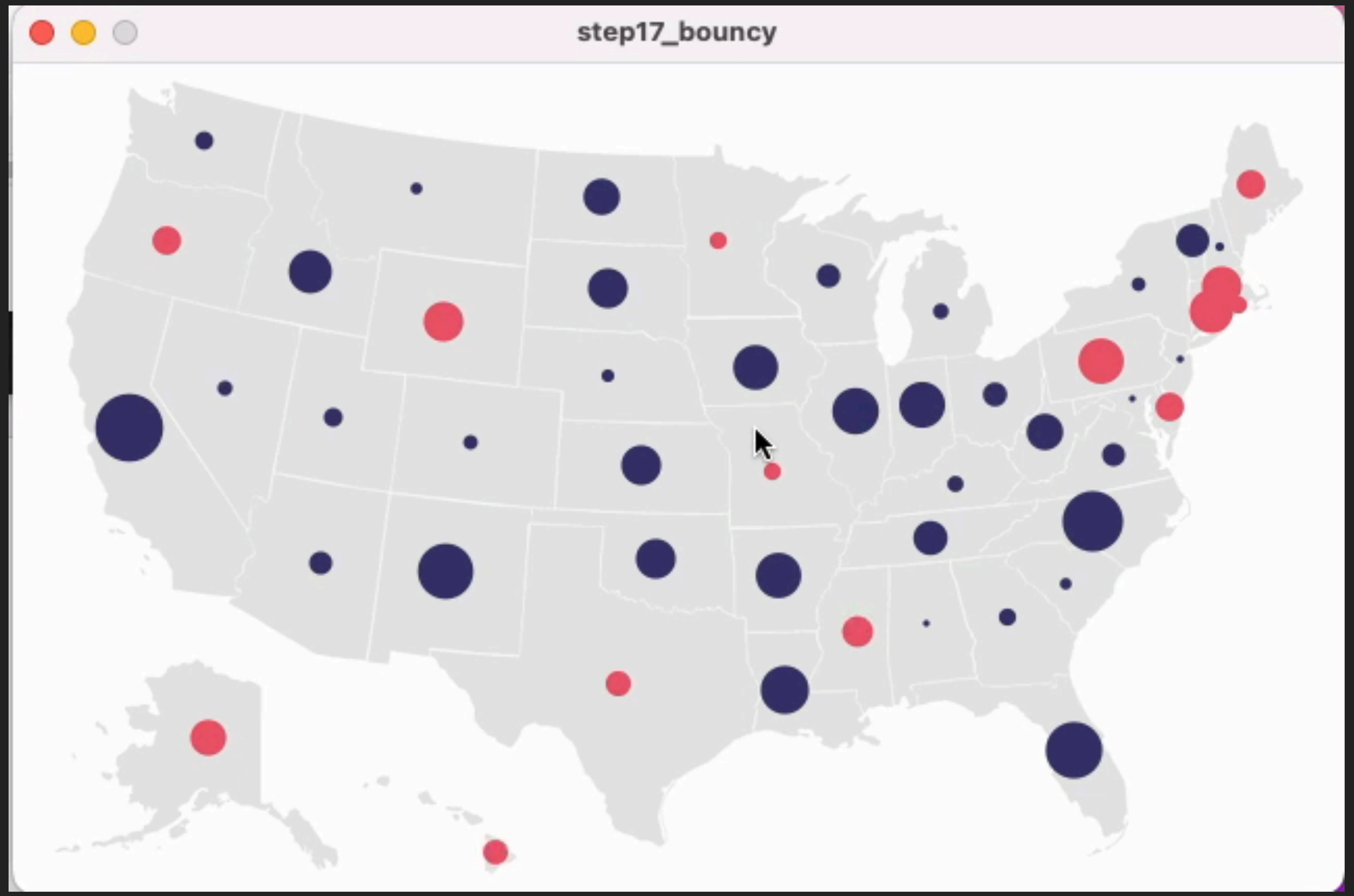
EXAMPLE

Integrator class used throughout Ben Fry's first Processing text

```
Integrator[] interpolators;
//. . .

void setup() {
  //. . .
  interpolators = new Integrator[rowCount];
  for (int row = 0; row < rowCount; row++) {
    float initialValue = dataTable.getFloat(row, 1);
    interpolators[row] = new Integrator(initialValue, 0.5, 0.01);
  }


void draw() {
  //. . .
  for (int row = 0; row < rowCount; row++) {
    interpolators[row].update();
  }
```



The processing sketch maintains an array of Integrators, and calls update on each one in the draw loop. Marks are rendered based on the integrators' current value, and when an integrator's target is changed, the mark animates to the new value.

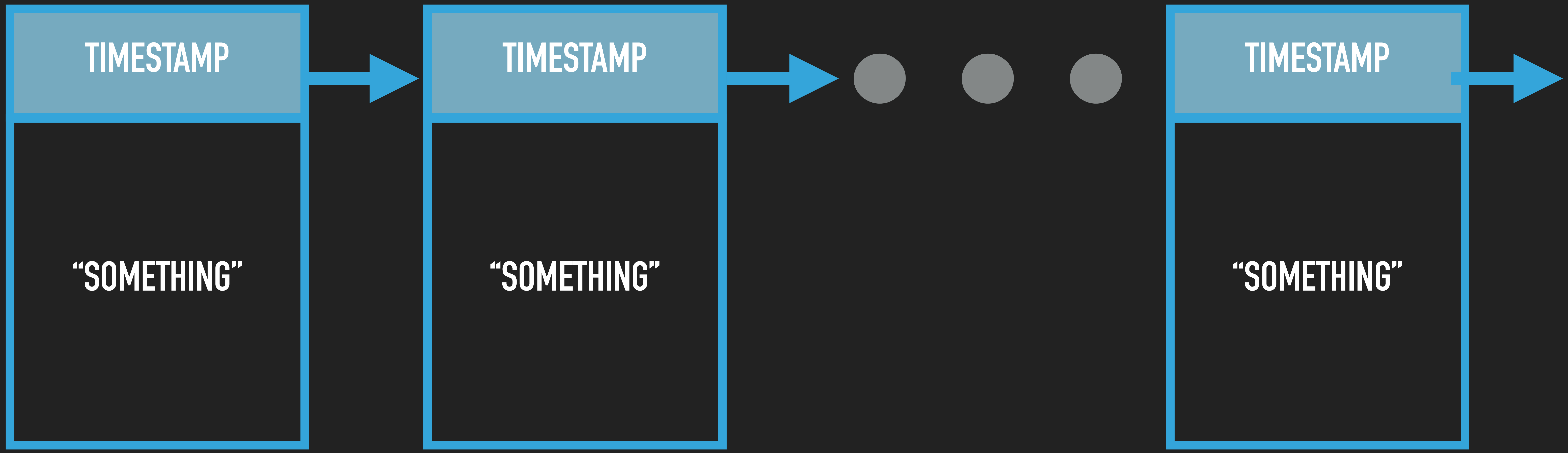
TIMELINES

MINIMAL TIMELINE ALGORITHM

What time is it NOW?

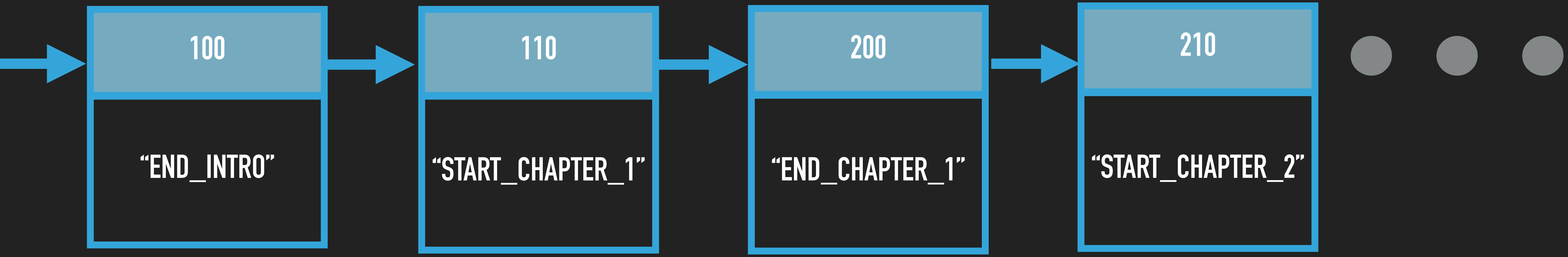
Based on that, what should I do?

MINIMAL TIMELINE ALGORITHM



"Something" could really be anything: a function to call, a variable with a new value, a string label...

MINIMAL TIMELINE ALGORITHM



Maintain list of events ordered by increasing timestamp.

Each time through a loop, **get the current time**.

Get a **list of any events with a timestamp < current time**.

Handle each event in the list. This could be as simple as a switch case statement with a list of modes.

```
//basic event handler:
switch(event_label) {

  case “END_INTRO”:
  //something
  break;

  case “START_CHAPTER_!”:
  //something
  break;

  case “END_CHAPTER_1”:
  //something
  break;

  //Etc…
}
```

MINIMAL TIMELINE ALGORITHM

Creating a case for each event does not scale well with more than a few events. A better system would have a flexible means of triggering just about anything possible in an environment: calling a function, updating a variable value, starting an easing that updates a variable value, triggering another timeline, etc.

BETTER TIMELINE ALGORITHM

Greensock - the same library with robust easing tools - has comprehensive timeline features, too:

```
var x=20, y=20, opacity=.5; //GS can manipulate variables (in "this" scope)
var obj = {prop: 10}; //GS can manipulate object fields and pretty much anything else
var tl = gsap.timeline({repeat: 20, repeatDelay: 1}); //An empty timeline,
                        //to hold events defined in setup
```

```
  //Add animation events to the timeline
  //See https://greensock.com/docs/v3/GSAP/Timeline
  tl.to(this, {x: 300, duration: 1}); //animate the X global to value 100 over 1 second
  tl.to(this, {y: 3*height/4, duration: 2, delay: -.5}); // animate y
  tl.to(this, {opacity: 255, duration: 1}); //animate opacity
```