**“EXPRESSIONS FOR AFTER EFFECTS”**

<https://readthedocs.org/projects/after-effects-expressions-guide/downloads/pdf/latest/>

<https://ukramedia.com/useful-expressions/>

<https://www.schoolofmotion.com/tutorials/amazing-expressions-in-after-effects>

**0.- CREATIVE TEXT ANIMATION (Animate - Position)**

freq = 1;

decay = 25;

duration = 0.30;

retard = textIndex\*thisComp.frameDuration\*1;

t = time - (inPoint + retard);

startVal = [100,100,100];

endVal = [0,0,0];

if (t < duration){

linear(t,0,duration,startVal,endVal);

}else{

amp = (endVal - startVal)/duration;

w = freq\*Math.PI\*2;

endVal + amp\*(Math.sin((t-duration)\*w)/Math.exp(decay\*(t-duration))/w);

}

**1.- Kinetic Growing Text (Expressions - Scale, Rotation, Position, Opacity)**

delay = .1;

myDelay = delay\*textIndex;

t = (time - inPoint) - myDelay;

if (t >= 0){

freq =3;

amplitude = 50;

decay = 8.0;

s = amplitude\*Math.cos(freq\*t\*2\*Math.PI)/Math.exp(decay\*t);

[s,s]

}else{

value

}

**2.- Kinetic Falling Text Animation (Expressions - Scale, Rotation, Position)**

delay = .1;

myDelay = delay\*textIndex;

t = (time - inPoint) - myDelay;

if (t >= 0) {

freq =3;

amplitude = 150;

decay = 8.0;

s = amplitude\*Math.cos(freq\*t\*2\*Math.PI)/Math.exp(decay\*t);

[s,s]

}

else {

value

}

**3.- Swinging Text Animation (Enable Per Character 3D, Rotation)**

swings=1.5;

timeOffset=0.50;

v=valueAtTime(time-timeOffset\*textIndex/textTotal)[0];

anim=2\*Math.PI\*v/100;

amp=easeOut(v,0,100,100,0);

Math.cos(anim\*swings)\*amp;

**4.- Bounce (method 1).**

First create an animation (ie. scale 100 to 120% or position from left to right). Next apply bellow’s expression to the property you’ve key-framed to make the animation bounce.

**n = 0;**

**if (numKeys > 0){**

**n = nearestKey(time).index;**

**if (key(n).time > time){**

**n--;**

**}**

**}**

**if (n == 0){**

**t = 0;**

**}else{**

**t = time - key(n).time;**

**}**

**if (n > 0 && t < 1){**

**v = velocityAtTime(key(n).time - thisComp.frameDuration/10);**

**amp = .06;**

**freq = 3;**

**decay = 5.0;**

**value + v\*amp\*Math.sin(freq\*t\*2\*Math.PI)/Math.exp(decay\*t);**

**}else{**

**value;**

**}**

**5.- Wiggle**

Can be applied to any property, but works great on the **position** to create a natural camera movement.

**wiggle(1,50);**

**6.- Squash and Stretch**

Apply it to the scale attribute of any object. Works best on icons or bullet points.

maxDev = 13; // max deviation in pixels

spd = 30; //speed of oscillation

decay = 1.0; //how fast it slows down

t = time - inPoint;

x = scale[0] + maxDev\*Math.sin(spd\*t)/Math.exp(decay\*t);

y = scale[0]\*scale[1]/x;

[x,y]

**7.- Motion Tail**

First animate the position of an object. Then apply bellow’s expression to the **position**:

//Apply to position

delay = 5; //number of frames to delay

d = delay\*thisComp.frameDuration\*(index - 1);

thisComp.layer(1).position.valueAtTime(time - d)

//Apply to Opacity

opacityFactor = .75;

Math.pow(opacityFactor,index - 1)\*100

**8.- Timer Up or Down**

Create a text layer and apply the expression to the **source text** attribute.

**//Define time values**

**var hour = Math.floor((time/60)/60);**

**var min = Math.floor(time/60);**

**var sec = Math.floor(time);**

**var mili = Math.floor(time\*60);**

**// Cleaning up the values**

**if (mili > 59){ mili = mili - sec\*60; }**

**if (mili < 10){ mili = "0" + mili; } if (sec > 59){ sec = sec - min\*60; }**

**if (sec < 10){ sec = "0" + sec; } if (min >= 59){ min = min - hour\*60; }**

**if (min < 10){ min = "0" + min; }**

**// no hour cleanup**

**if (hour < 10){ hour = "0" + hour; }**

**//Output**

**hour + ' : ' + min + ' : ' + sec + ' : ' + mili;**

**9.- SELF RESIZING BOX TEXT**

Revisa éste tutorial: <https://motionarray.com/learn/after-effects/autoscale-text-after-effects/>

POSITION EXPRESSION

s=thisComp.layer("Text01");

w=s.sourceRectAtTime().width/2;

h=s.sourceRectAtTime().height/2;

l=s.sourceRectAtTime().left;

t=s.sourceRectAtTime().top;

[w+l, h+t]

SIZE EXPRESSION

s=thisComp.layer("Text01");

w=s.sourceRectAtTime().width;

h=s.sourceRectAtTime().height;

[w,h]

**10.- BOUNCE (method 2)**

Apply in Position.

// Ease and Wizz 2.5.3 : outBounce : All keyframes

// Ian Haigh (http://aescripts.com/ease-and-wizz/)

// Last built: 2017-08-08T09:11:05+10:00

function easeandwizz\_outBounce(t, b, c, d) {

if ((t/=d) < (1/2.75)) { return c\*(7.5625\*t\*t) + b }

else if (t < (2/2.75)) { return c\*(7.5625\*(t-=(1.5/2.75))\*t + .75) + b }

else if (t < (2.5/2.75)) { return c\*(7.5625\*(t-=(2.25/2.75))\*t + .9375) + b }

else { return c\*(7.5625\*(t-=(2.625/2.75))\*t + .984375) + b }

}

function easeAndWizz() {

var n = 0;

if (numKeys > 0) {

n = nearestKey(time).index;

if (key(n).time > time) { n-- }

}

try {

var key1 = key(n);

var key2 = key(n+1);

} catch(e) {

return null;

}

// determine how many dimensions the keyframes need

var dim = 1; // It's gotta have at least ONE dimension

try {

key(1)[1];

dim = 2;

key(1)[2];

dim = 3;

} catch(e) {}

t = time - key1.time;

d = key2.time - key1.time;

sX = key1[0];

eX = key2[0] - key1[0];

if (dim >= 2) {

sY = key1[1];

eY = key2[1] - key1[1];

if (dim >= 3) {

sZ = key1[2];

eZ = key2[2] - key1[2];

}

}

if ((time < key1.time) || (time > key2.time)) {

return value;

} else {

val1 = easeandwizz\_outBounce(t, sX, eX, d);

switch (dim) {

case 1:

return val1;

break;

case 2:

val2 = easeandwizz\_outBounce(t, sY, eY, d);

return [val1, val2];

break;

case 3:

val2 = easeandwizz\_outBounce(t, sY, eY, d);

val3 = easeandwizz\_outBounce(t, sZ, eZ, d);

return [val1, val2, val3];

break;

default:

return null;

}

}

}

(easeAndWizz() || value);

**11.- AUTOMATIC FADE**

Apply in Opacity.

transition = 20;

if (marker.numKeys<2){

tSecs = transition / ( 1 / thisComp.frameDuration);

linear(time, inPoint, inPoint + tSecs, 0, 100)

- linear(time, outPoint - tSecs, outPoint, 0, 100)

}else{

linear(time, inPoint, marker.key(1).time, 0, 100)

- linear(time, marker.key(2).time, outPoint, 0, 100)

}

**12.- SQUASH AND STRETCH**

This bouncing squash and stretch expression should make your animation a bit more alive by adding a proportional scale to your shapes or images.

maxDev = 13; // max deviation in pixels

spd = 30; //speed of oscillation

decay = 1.0; //how fast it slows down

t = time - inPoint;

x = scale[0] + maxDev\*Math.sin(spd\*t)/Math.exp(decay\*t);

y = scale[0]\*scale[1]/x;

[x,y]

**13.-LOOPING WIGGLE (by Dan Ebberts)**

The looping [wiggle expression](https://blog.motionisland.com/wiggle-expression-after-effects-tutorial/) become handy when you want an object to move randomly in your composition. Looping wiggle should be seamless and you shouldn’t see a cut, this expression is also used for background animation.

freq = 1;

amp = 110;

loopTime = 3;

t = time % loopTime;

wiggle1 = wiggle(freq, amp, 1, 0.5, t);

wiggle2 = wiggle(freq, amp, 1, 0.5, t - loopTime);

linear(t, 0, loopTime, wiggle1, wiggle2)

14.- TIME

## Time expression is good for constant animation, the higher the number next to “time” below the faster your animation will be.

time\*150

**15.- ROTATE MULTIPLICATION**

Rotate Multiplication is my favorite expression, basically it allows you to rotate multiple shapes around a center point and define how many shape you want this case below is 20 shapes layers until it make a circle. The rotate mutiplication is awesome to create circle burst animation.

index\*360/20



**16.- Constant Rotation Per Second**

Here is another After Effects rotation expression, no key frames needed to make a simple 360 rotation on a layer.

veloc = 360; //360 Degree Rotation per Second

r = rotation + (time - inPoint) \*veloc;

[r]

**17.- Move Object X Pixel per Second**

If your tired of making key frames, you’ll like this expression it allows you to move an object on the x axis per second.

veloc = 150; //Move object 150 pixel on x axis per second

x = position[0] + (time - inPoint) \*veloc;

y = position[1];

[x,y]

**MOVE OBJECT Y PIXEL PER SECOND**

veloc = 150; //Move object 150 pixel on y axis per second

x = position[0];;

y = position[1] + (time - inPoint) \*veloc;

[x,y]

**17.- Scale Constantly per Second**

veloc = 1; //horizontal velocity (pixels per second)

x = scale[0] + (time - inPoint) \*veloc;

y = scale[0] + (time - inPoint) \*veloc;

[x,y]

**18.- Inertial Bounce Expression**

The Inertial Bounce expression is a bit more complex than Wiggle, but when used properly, it can be really helpful in simulating physics for a bounce animation. For example, if you wanted to have a ball fall from the top of the screen, hit the bottom, and bounce with decaying force until it stopped, you could do this is individual keyframes and the graph editor. But it would take a long time to get right.

n = 0;

if (numKeys > 0){

n = nearestKey(time).index;

if (key(n).time > time){

n--;

}

}

if (n == 0){

t = 0;

}else{

t = time - key(n).time;

}

if (n > 0){

v = velocityAtTime(key(n).time - thisComp.frameDuration/10);

amp = .05;

freq = 4.0;

decay = 2.0;

value + v\*amp\*Math.sin(freq\*t\*2\*Math.PI)/Math.exp(decay\*t);

}else{

value;

}

**19.- Falling Leaf (3D)**

If you edit the parameters, you need to make sure that the oscillating

frequency is the same in both the position and y-rotation expressions (or map

it to a slider control). If you wanted to apply this to a bunch of leaves,

you'd want to introduce some randomness into the various parameters along with

a random starting phase.

**POSITION:**

yVelocity = 200; //pixels per second

oscFreq = 1.5; //oscillations per second

oscDepth = 35; //oscillation depth (pixels)

drift = 25; // drift (wind?) (pixels per second: - = left, + = right)

value + [oscDepth\*Math.sin(oscFreq\*Math.PI\*2\*time) + drift \*time,

yVelocity\*time,0]

**Z ROTATION:**

seed\_random(index,true);

random(360);

**Y ROTATION:**

oscFreq = 1.5;

maxTilt = 15; //degrees

maxTilt\*Math.cos(oscFreq\*Math.PI\*2\*time)

**20.- Make a layer revolve in a circle**

You can create an expression without using properties from other layers. For example, you can make a layer revolve in a perfect circle. Select a layer, press P to reveal its Position property in the Timeline panel, and Alt-click (Windows) or Optionclick (Mac OS) the stopwatch to the left of the property name.

[(thisComp.width/2), (thisComp.height/2)] + [Math.sin(time)\*50, -Math. ˓→cos(time)\*50]

**21.- Rotate the hands of a clock**

1. Import or create two long, narrow solid-color layers: an hour hand and a minute hand.

2. Set the anchor points at the ends of the layers.

3. Move the layers so that the anchor points are at the center of the composition.

4. Set Rotation keyframes for the hour hand.

5. Select the Rotation property for the minute hand and choose Animation > Add Expression.

6. Drag the pick whip to the Rotation property for the hour hand. The following expression appears:

thisComp.layer("hour hand").rotation

7.- To make the minute hand rotate 12 times as fast as the hour hand, add \* 12 at the end of the expression as follows:

thisComp.layer("hour hand").rotation \* 12

**22.- Position one layer between two others**

1. Start with three layers.

2. Animate the positions of the first two layers in the Timeline panel.

3. Select the third layer, press P to reveal the Position property, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch button to the left of the property name.

4. Enter the following in the expression field:

(thisComp.layer(1).position + thisComp.layer(2).position)/2

**23.- Create a trail of images**

1. Start with two solid-color layers that are scaled to approximately 30% of the composition size. (See Solid-color layers and solid-color footage items.)

2. Animate the position of the first layer.

3. Select the second layer, press P to reveal the Position property, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch button to the left of the property name.

4. Enter the following in the expression field:

thisComp.layer(thisLayer, -1).position.valueAtTime(time - .5)

5. Duplicate the last layer five times by selecting it and pressing Ctrl+D (Windows) or Command+D (Mac OS) five times.

\*All layers follow the same path, and each is delayed 0.5 seconds from the previous.

**24.- Start or stop wiggle at specific time**

You can use any expression in place of the wiggle expression used here, to begin and end the influence of any expression at a specific time.

Apply the following expression to a property to wiggle it beginning at time 2 seconds:

timeToStart = 2; if (time > timeToStart) { wiggle(3,25); } else { value; }

Apply the following expression to a property to stop wiggling it at time 4 seconds:

timeToStop = 4; if (time > timeToStop) { value; } else { wiggle(3,25); }

Apply the following expression to a property to start wiggling it at time 2 seconds and stop wiggling it at time 4 seconds:

timeToStart = 2;

timeToStop = 4;

if ((time > timeToStart) && (time < timeToStop)) { wiggle(3,25); } else { value; }

**25.- Fade opacity of a 3D layer based on distance from camera**

Apply the following expression to the Opacity property of a 3D layer:

startFade = 500; // Start fade 500 pixels from camera.

endFade = 1500; // End fade 1500 pixels from camera.

try { // Check whether there's a camera C = thisComp.activeCamera.toWorld([0,0,0]); } catch (err) { // No camera, so assume 50mm w = thisComp.width \* thisComp.pixelAspect; z = (w/2)/Math.tan(degreesToRadians(19.799)); C = [0,0,-z]; }

P = toWorld(anchorPoint);

d = length(C,P);

linear(d,startFade,endFade,100,0)

The fade starts at a distance of 500 pixels from the camera and is complete at 1500 pixels from the camera. The linear interpolation method is used to map distance values to opacity values.