

UNDERSTANDING DOOGLUS' ZDEPTH ANIMATION: "BALLS"

INTRODUCTION

Dooglus created a cool animation using a complex conversion and link of the Z depth parameter and the x and y positions of the canvases. I want to try to understand how he made the animation and explain it to you.

In the animation there are 7 beveled and blurred balls that are placed as if they were in a perspective circle. Also the balls turn synchronized together with variable speed.

See the animation and sifz file here: <http://es.youtube.com/watch?v=YTpSfUthuVE>

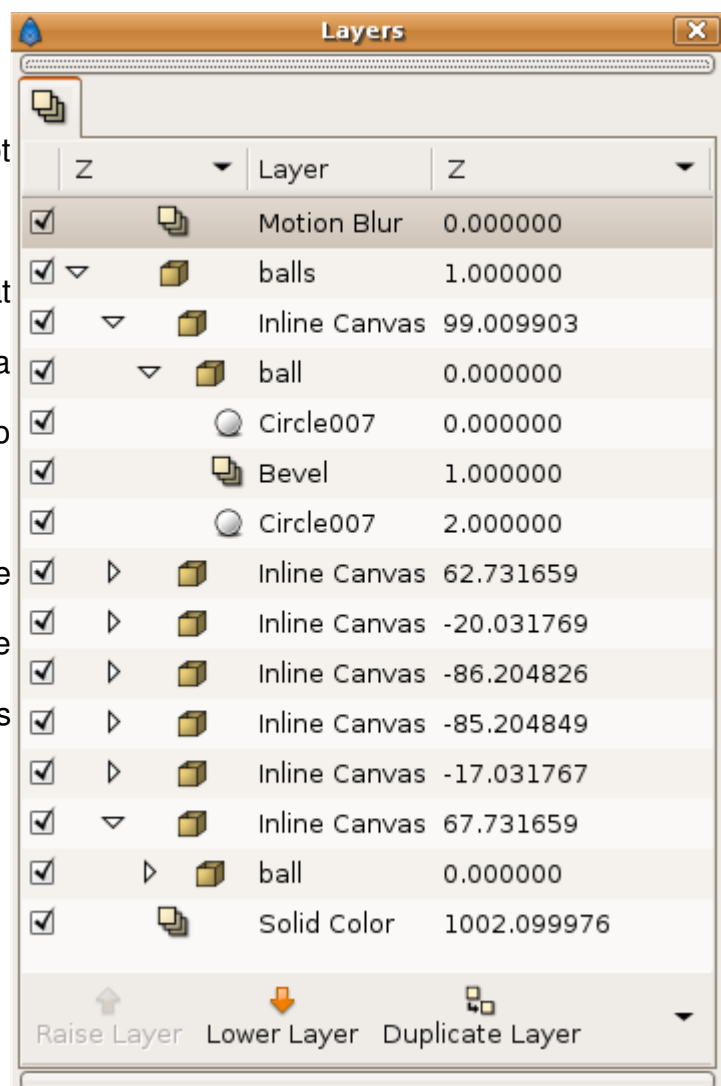
Let's have a look at the layers he used:

You can see that there are three root layers: Motion blur, balls and solid color.

The balls layer is a paste canvas that holds the 7 inline canvases, each with a ball inside. Each ball is composed of two circles and a Bevel layer in the middle.

Also notice that, besides its color, the main difference between the inline canvas for the balls is that the Z depth is different for each one.

That will be explained later.

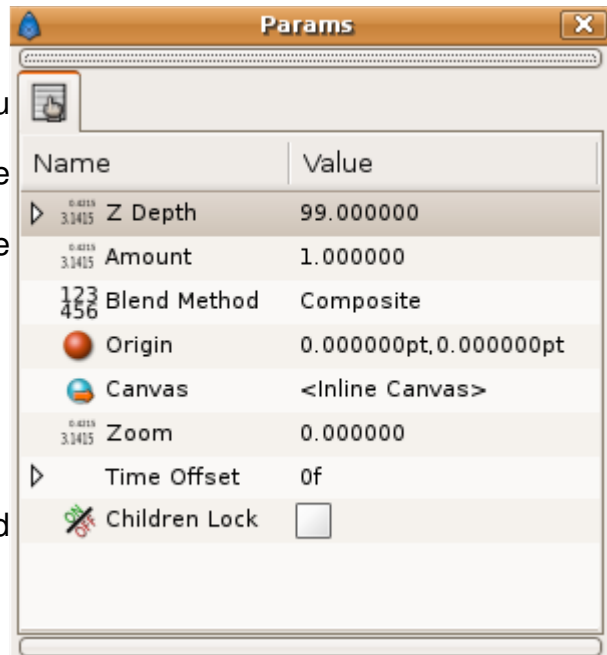


THE PARAMETERS

When you select any of the inline canvas you can see that two of the common parameters are converted to some other type. Let's explore the Z depth first and later the Time Offset.

1. Z Depth:

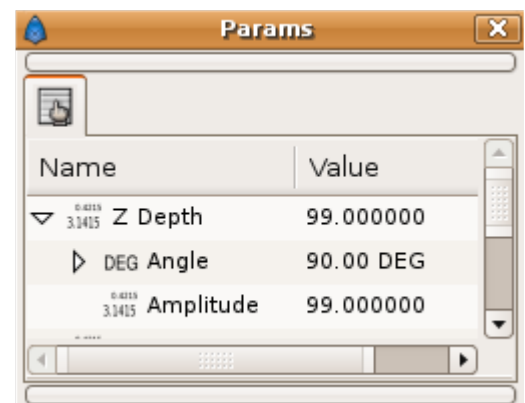
The first thing we can see is that he converted the Z depth to a sine type:



$$Z \text{ depth} = \text{Amplitude} * \text{Sin}(\text{Angle})$$

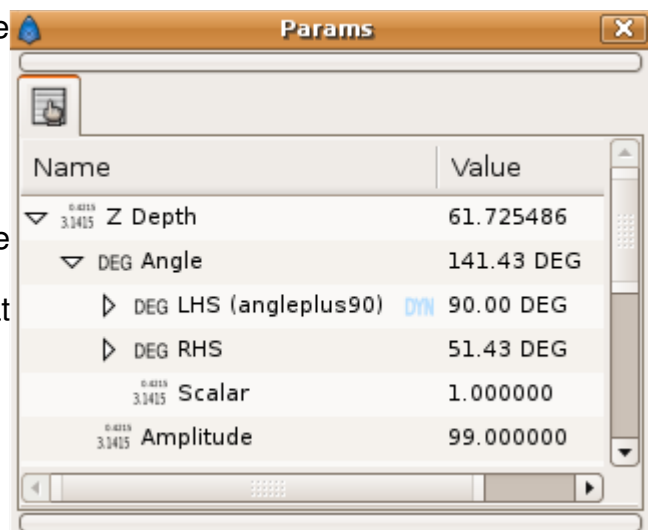
but the Angle is also a converted parameter:

$$\text{Angle} = (\text{LHS} + \text{RHS}) * \text{Scalar} = \text{LHS} + \text{RHS}$$



In all the inline canvases the Scalar is the same (1) so we can ignore it.

Again we can see that RHS and LHS are converted parameters so let's see what hides inside them.



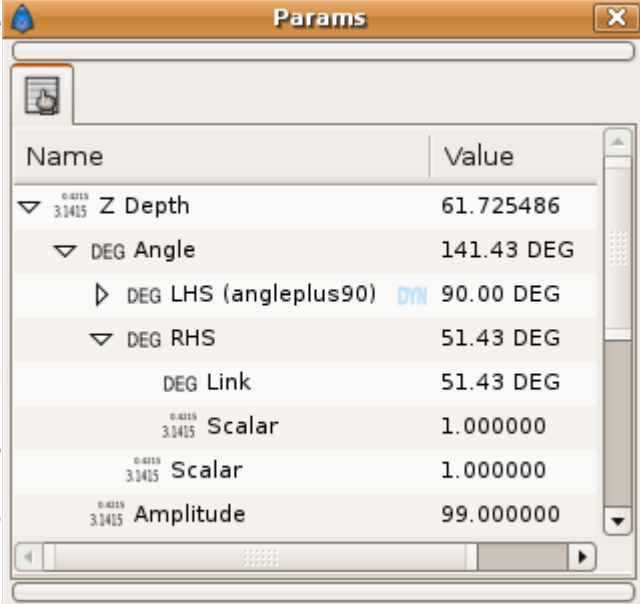
Let's start with RHS; that is the easiest.

If we open it for the different layers we can see that is not the same for every one. Let's open the second layer in the list:

This RHS value has a conversion of a Scale
convert type:

$$\text{RHS} = \text{Link} * \text{Scalar}$$

In this case Scalar's value is 1.0 but if you look to the other layers you can see that this value is going from 0 to the first layer to 6 the last layer.



Name	Value
Z Depth	61.725486
DEG Angle	141.43 DEG
DEG LHS (angleplus90) DYN	90.00 DEG
DEG RHS	51.43 DEG
DEG Link	51.43 DEG
Scalar	1.000000
Scalar	1.000000
Amplitude	99.000000

If you Look to the Link value (51.43) it is exactly the same than $360/7$. This is not casual. Using this technique applies a rotation to each layer of $360/7 * \text{Scale}$, being Scale 0,1,2,3,4,5 and 6 for each layer and then each layer is rotated $360/7$ from the next.

Let's look at LHS now:

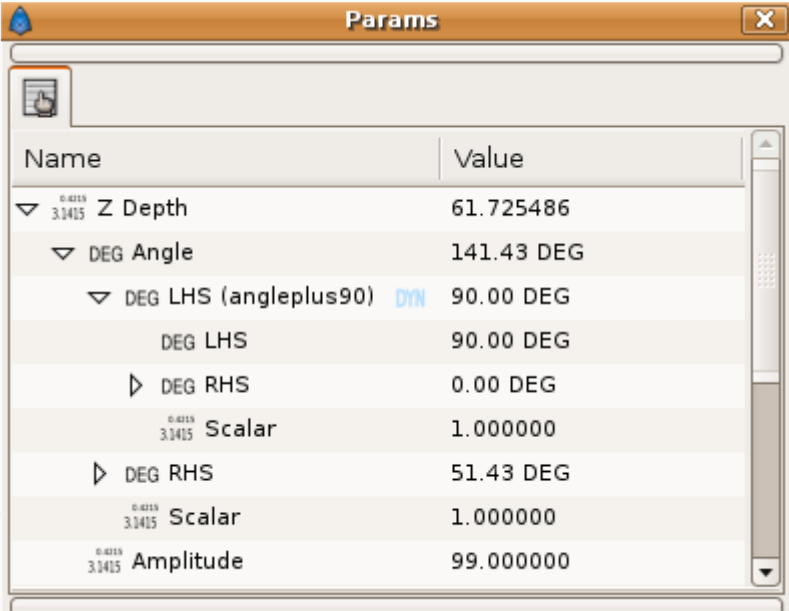
LHS is the addition of two values
again:

$$\text{LHS} = \text{LHS}' + \text{RHS}'$$

(I'll add a quote to the new
parameters to distinguish them

from the others). In this case we

can ignore Scalar again because it is always = 1. You can see that LHS is linked to "angleplus90". This means that the exported RHS' value (0.0 DEG in the sample) is the main



Name	Value
Z Depth	61.725486
DEG Angle	141.43 DEG
DEG LHS (angleplus90) DYN	90.00 DEG
DEG LHS	90.00 DEG
DEG RHS	0.00 DEG
Scalar	1.000000
DEG RHS	51.43 DEG
Scalar	1.000000
Amplitude	99.000000

angle and LHS' only adds 90 DEG to it to obtain another angle from the first.

Let's open RHS' and see what it has inside:

In this case the RHS' parameter is converted to a Linear value:

$$\text{RHS}' = \text{Rate} \times \text{time} + \text{Offset}$$

Although Offset is exported it doesn't do anything because it is =0 always. Maybe he forgot to Disconnect it.

So here we can see the engine of the animation: The RHS' is the one who is going to change over the time at a Rate of 90 DEG per second. Let's combine all the above equations to see how the Z depth looks:

$$\text{Z depth} = \text{Amplitude} \times \sin(\text{Angle})$$

$$\text{Angle} = \text{LHS} + \text{RHS}$$

$$\text{RHS} = \text{Link} \times \text{Scalar}$$

$$\text{LHS} = \text{LHS}' + \text{RHS}'$$

$$\text{RHS}' = \text{Rate} \times \text{time} + \text{Offset}$$

And these values are constant

$$\text{Amplitude} = 99.0$$

$$\text{LHS}' = 90 \text{ DEG}$$

$$\text{Scalar} = 0, 1, 2, 3, 4, 5, 6 \text{ (for each layer)}$$

$$\text{Zdepth} = 99.0 \times \sin \left(90^\circ + 90^\circ / \text{sec} \times \text{time}(\text{sec}) + \frac{360}{7} \times \text{Scalar} \right)$$

Remember that this formula only gives the Z-depth value of the layer (they are at the same

position unless something else changes it)

2. Time OffSet

The Time OffSet parameter is converted to a Linear value so it gives two new sub-parameters:

$$\text{TimeOffset} = \text{Rate} * \text{Scalar}$$

Where:

Rate = 16 f, and

Scalar = 0,1,2,3,4,5,6 for each layers.

That means that, while the Z depth that is calculated based on its rotation angle, each layer is time shifted 16 frames per second on its internal movement. So you see first layer $16f*0$ later, the second $16f*1$ later, the third $16f*2$ later and so on.

This will affect to the animation of the ball inside the each layer.

16 frames was not selected arbitrary. If you multiply $16*7$ layers then you achieve 112 frames. Taking account that the frame speed is 28 fps then a ball repeats its position after 4 seconds ($112/28=4$). Also remember that rotation rate is 90 DEG per second, which gives 360 DEG per 4 seconds.

Let's see how he positions the ball in the X and Y axis.

3. Position

To see the position of the balls we will go to the ball canvas for each ball inside the inline

canvas (see the first image). Let's see all the parameters and their conversions in one shot:

Name	Value
Z Depth	0.000000
Amount	1.000000
Blend Method	Composite
Origin	0.000000pt,59.999998pt
X-Axis	0.000000
DEG Angle	0.00 DEG
DEG Rate (rotation rate)	90.00 DEG
DEG Offset	0.00 DEG
DEG Link	51.43 DEG
Scalar	0.000000
Amplitude (x range)	2.000000
Y-Axis (yvalue)	1.000000
DEG Angle (angleplus90)	90.00 DEG
DEG LHS	90.00 DEG
DEG RHS	0.00 DEG
DEG Rate (rotation rate)	90.00 DEG
DEG Offset	0.00 DEG
Scalar	1.000000
Amplitude (y range)	1.000000
Canvas	<Inline Canvas>
Zoom	0.000000
Time Offset	0f
Children Lock	<input type="checkbox"/>

Here you can see that X is a sine converted parameter and the angle is a linear converted parameter of the “rotation rate” exported parameter. Again the Offset parameter doesn't do anything. The scaled parameter is an exported parameter with a value of 2 (the length of the longer axis of the ellipse). And the Y parameter has a similar conversion to a sine type but for a 90 DEG rotated angle. Taking account that $\sin(\alpha+90) = \cos(\alpha)$ then the formula becomes easy to understand.

Notice that you'll never know what was the original exported parameter because they are exactly the same. Also notice that the exported parameter carry on with all the exported sub parameters, as the angleplus90 one.

$$X = xrange \times \sin(\text{rotation rate})$$

$$y = yrange \times \sin(\text{angleplus90})$$

4. The variable speed

Now we have a set of seven balls, with a calculated Z depth each based on its rotation and a X and Y position based also in the rotation. As we have seen the rotation is 1 turn each 4 seconds.

But you can see in the animation that the rotation speed is changing. How did he do it?. So easy. Using another paste canvas that holds all the inline layers for each ball and adjusting the Time Offset parameter through time. In the file the Time OffSet parameter has the following waypoints:

Frame	Time OffSet	What you obtain in the animation
0s	0s	Static (the balls aren't static – the offset is only 0s for 1 frame)
2s	-5s	Accelerated animation in reverse
4s	-2s	Same but less speed

6s	14f	Accelerated animation in a forward direction
8s	2s 14f	Same but less speed
10s	4s	Same but more speed
12s	5s	Same but more speed
14s	4s	Same but less speed
16s	2s	Same but less speed
18s	-5s	Accelerated animation in reverse
20s	-20s	Same but more speed.

I hope my explanations didn't hurt your brain too much. I've enjoyed researching this cool, animation so much.

Enjoy!

Carlos

Oct 7, 2007

Thanks to dooglus (Chris Moore) for correct the writing style and deep explain some conversion and link concepts.