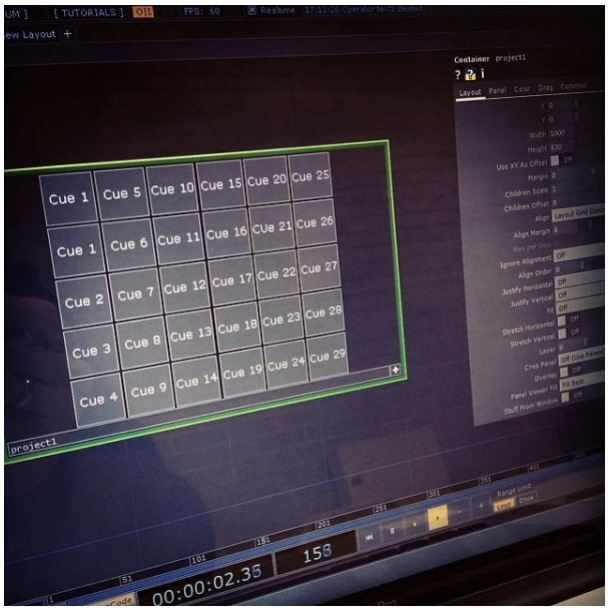
Matthew Ragan

Media Maker and Interactive System Designer

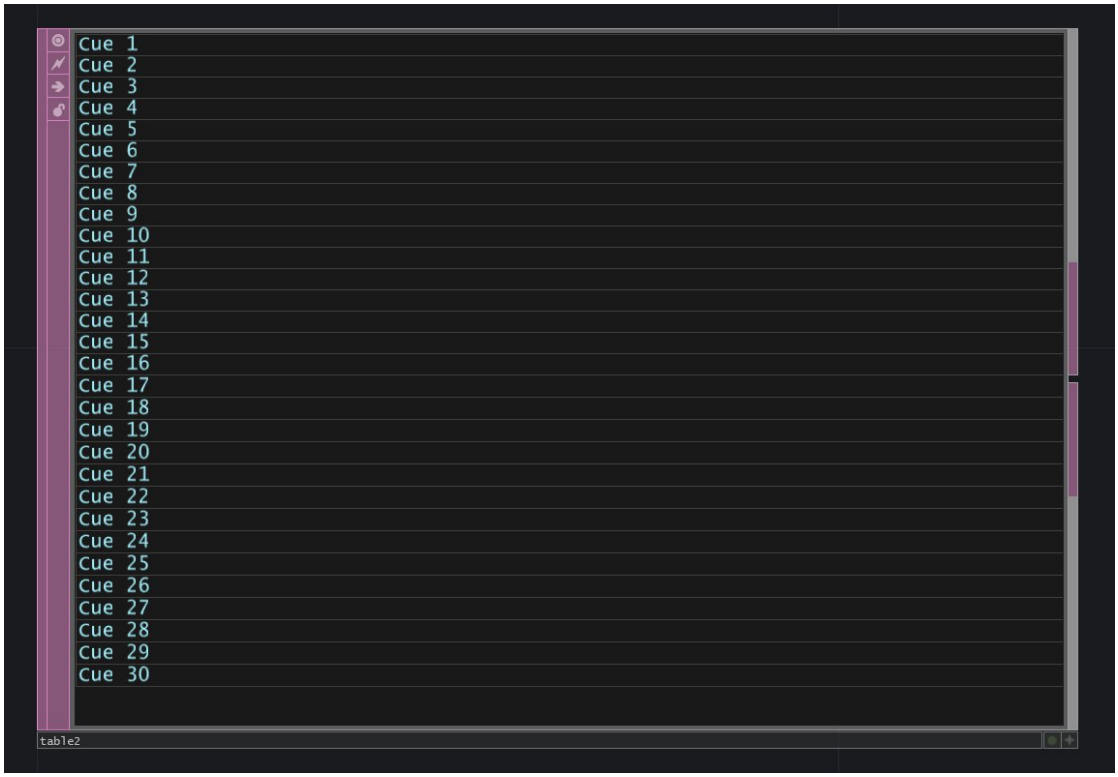
TouchDesigner | Replicators, and Buttons, and Tables, oh my



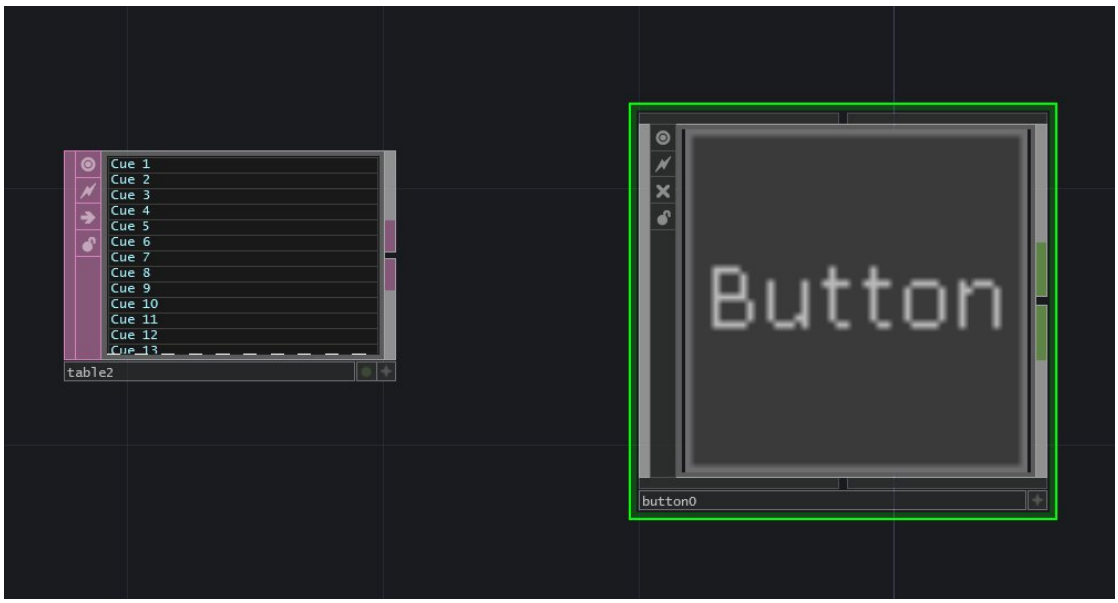
I want 30 buttons. Who doesn't want 30 buttons in their life? Control panels are useful for all sorts of operations in TouchDesigner, but they can also be daunting if you're not accustomed to how they work. In the past when I've wanted lots of buttons and sliders I've done all of my lay-out work the hard way… like the hardest way possible, one button or slider at a time. This is great practice, and for anyone who is compulsively organized this activity can be both maddening and deeply satisfying at the same time. If you feel best when you're meticulously aligning your buttons and sliders in perfect harmony, this might be the read for you. If, however, you like buttons (lots of them), and you want to be able to use one of the most powerful components in TouchDesigner, then Replicators might just be the tool you've been looking for – even if you didn't know it.

Replicators, big surprise, make copies of things. On the surface of it, this might not seem like the most thrilling of operators, but lets say that you want to automate a process that might be tedious if you set out to do it by hand. For example, let's say you're designing a tool or console that needs 30 buttons of the same type, but you'd like them have unique names and follow a template and have unique names. You could do this one button at a time, but if you ever change something inside of the button (like maybe you want the colors to change depending on their select state), making that change to each button might be a huge hassle. Replicators can help us make that process fast, easy, just plain awesome.

First thing, let's make a container and dive inside of it to start making our buttons. Next let's make a table. I'm going to use my table DAT to tell the replicator how to make the copies of my button. As you start to think about making a table imagine that each row (or column) is going to be used to create our copies. For example, I've created a table with thirty rows, Cue 1 – Cue 30.



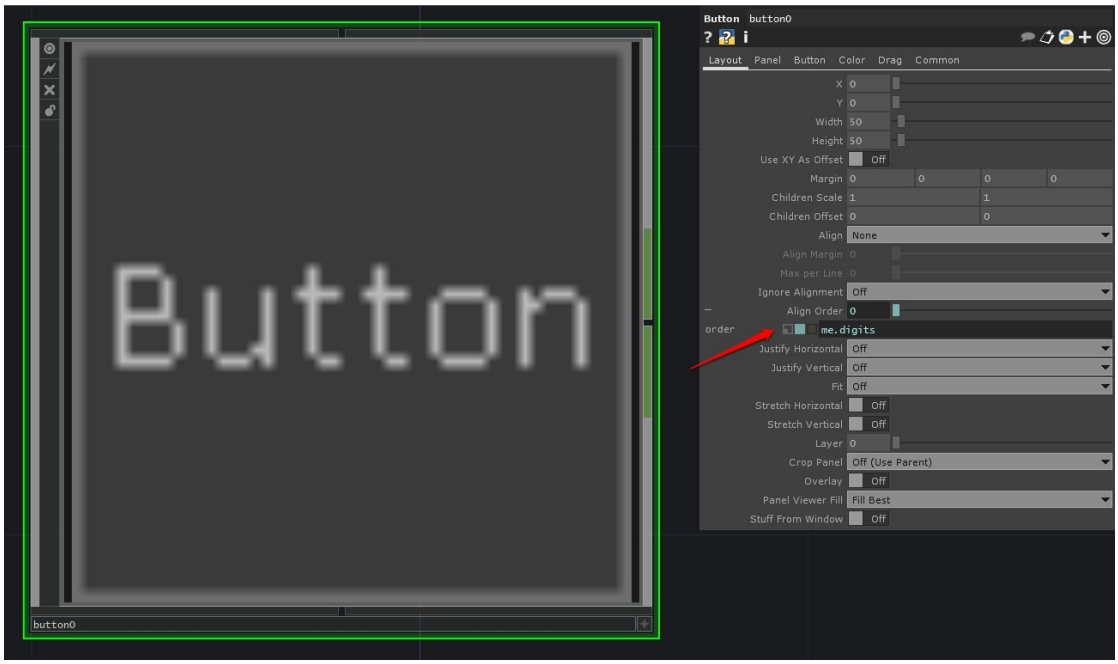
Next let's make a button COMP and rename it **button0** (don't worry, this will make sense in a a few minutes).



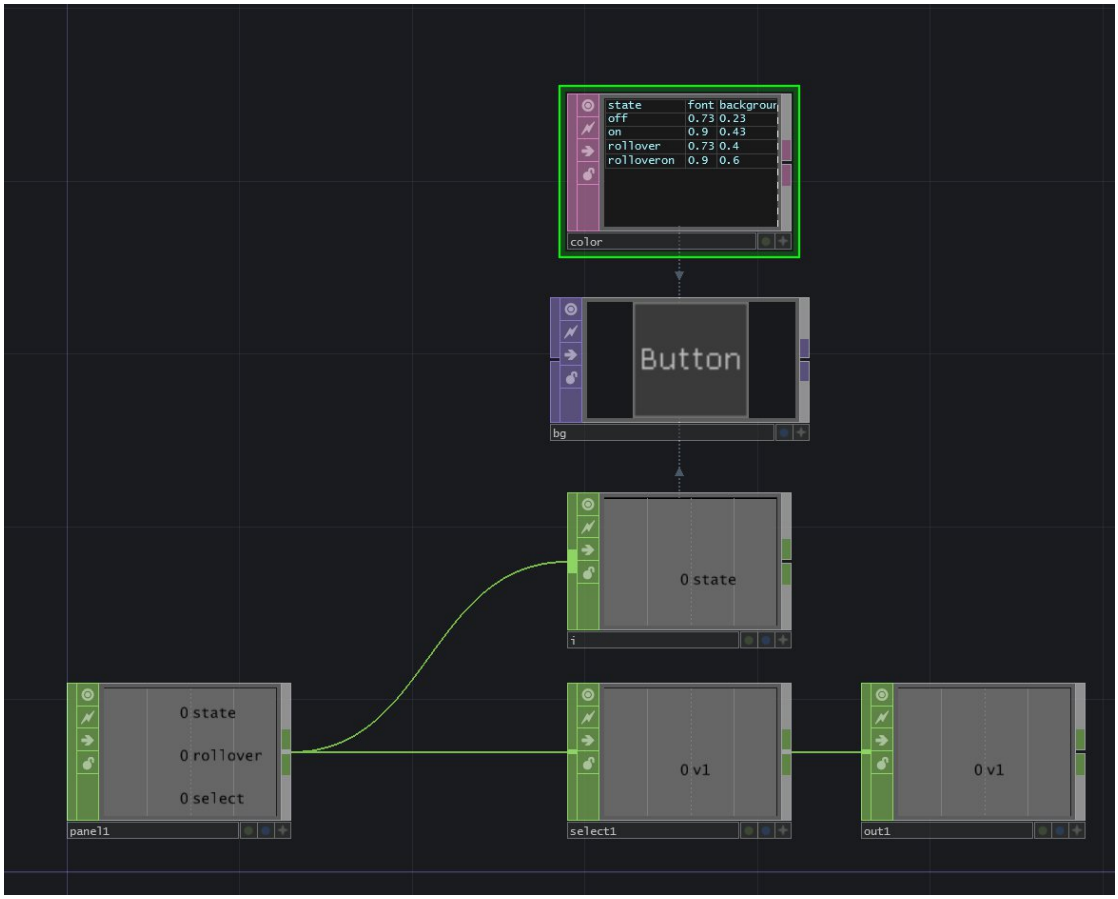
Alright, now we're gonna do some detailed work to set-up our button so some magic will happen later. First up we're going to make a small change to the Align Order of our button. The Align Order parameter of a button establishes the order in which your buttons are arranged in the control panel when you use the Align Parameter (if this doesn't make sense right now, hang in there and it will soon). We're going to use a simple python expression to specify this parameter. Expand the field for the Align Order and type the python expression:

me.digits

What's this all about? This expression is calling for the number that's in the name of our button. Remember that we renamed our button to **button0**. Using this expression we've told TouchDesigner that the alignment order parameter of this button should be **0** (in many programming languages you start counting at 0, so this makes this button the first in a list).



Alright, now let's start making some magic happen. Let's go inside of our button and make a few changes to the background text. If this is your first time looking inside of the button component take a moment to get your bearings – we have a table DAT that's changing the color of the button based on how we interact with this component, we have some text that we're using for the background of the button, and we have a few CHOPs so we can see what the state of our button is when it's toggled or pressed.



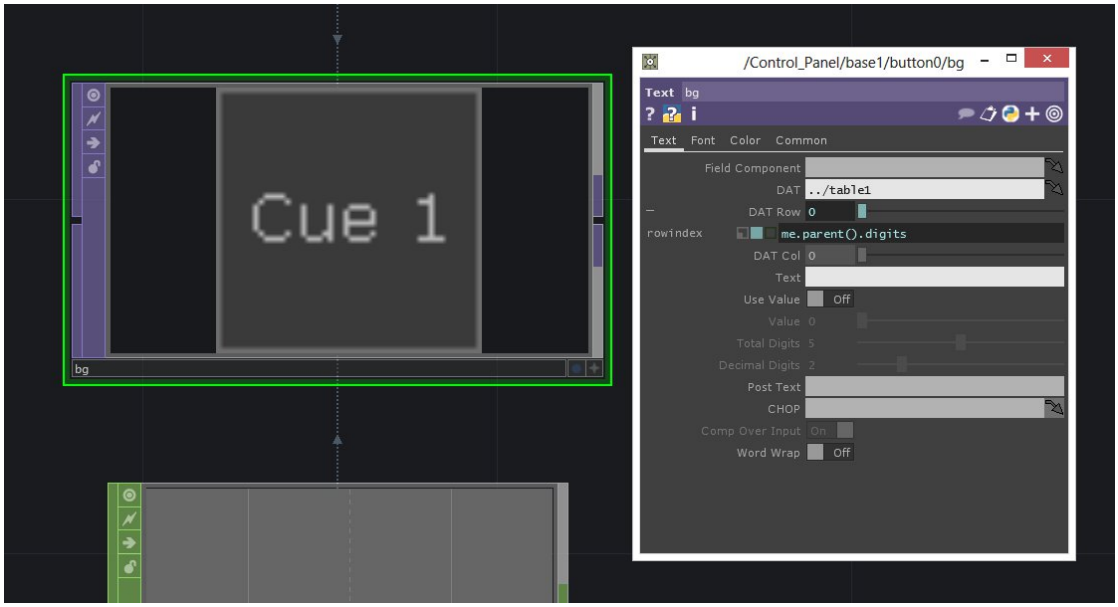
Next we're going to take a closer look at just the text TOP called **bg**. Looking first at the text page of the parameters dialogue gives us a sense of what's happening here, specifically we can see that the text field is where we can change what's displayed on this button. First up we're going to tell this TOP that we want to pull some data from a DAT. In the DAT field type the path:

../table1

Next change the DAT Row to the following python expression:

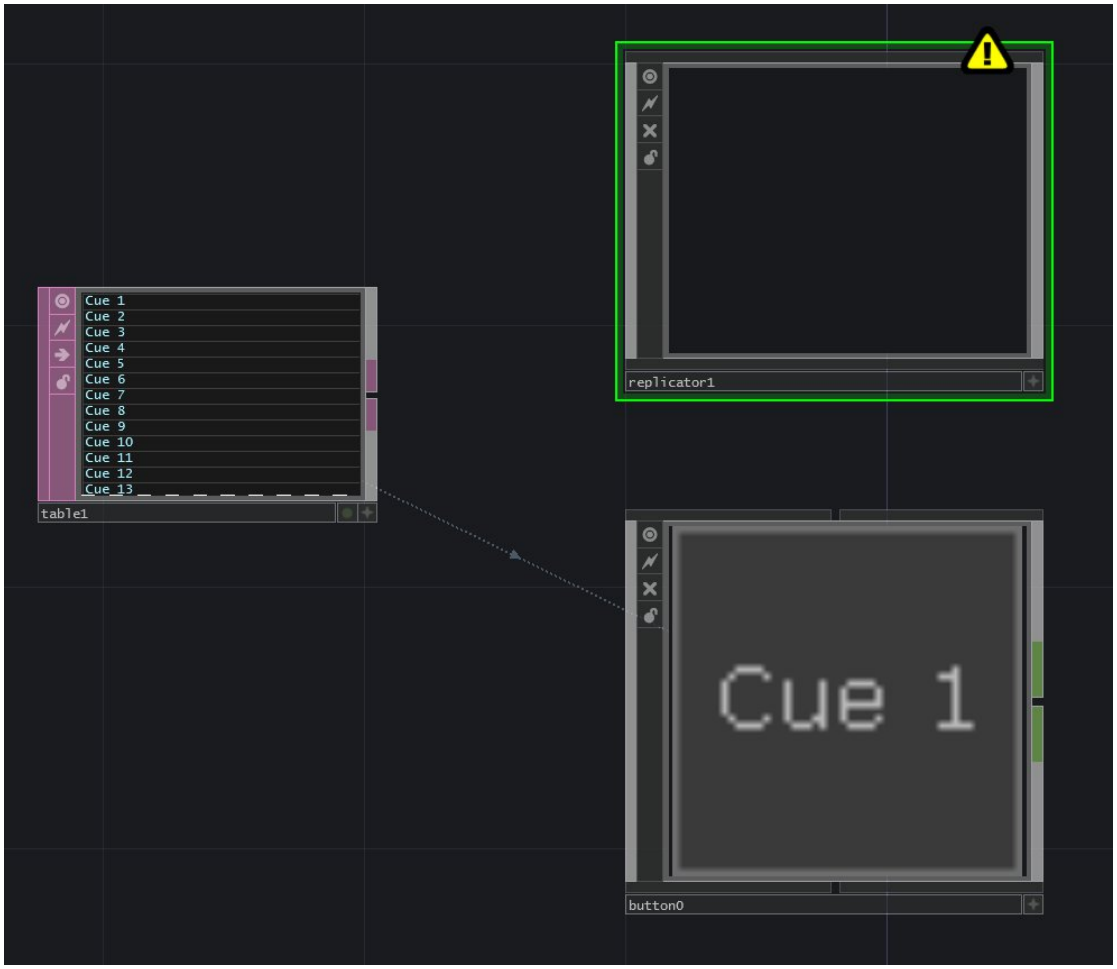
me.parent().digits

Finally, clear the text parameter so it's blank. With any luck you're button should now read, "Cue 1," or whatever you happened to type in the first row of your table.

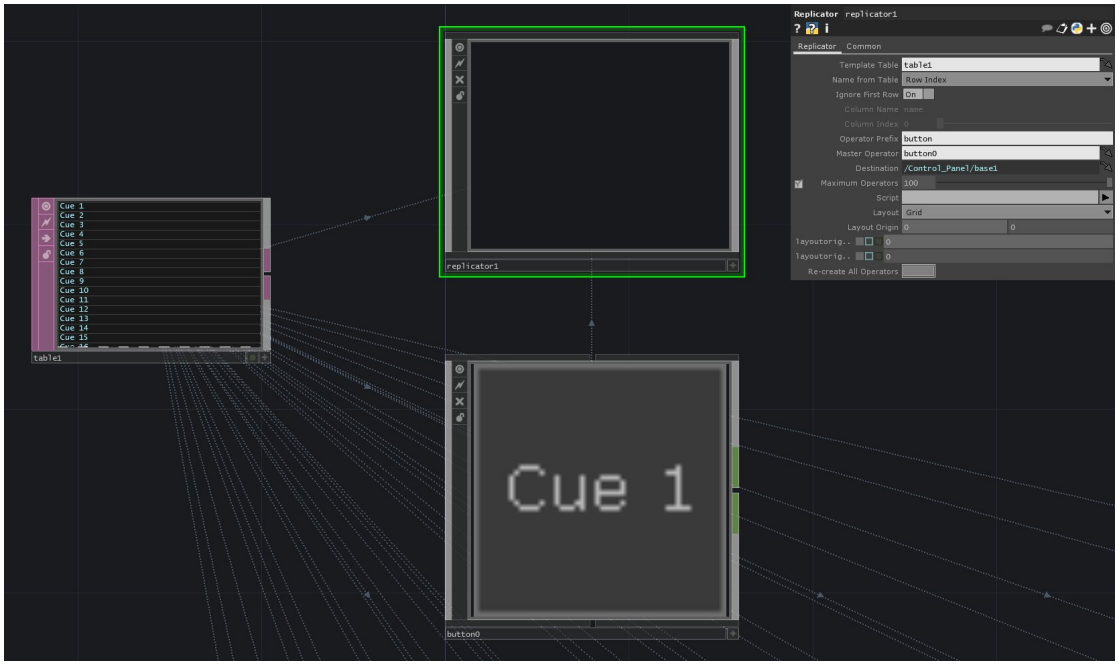


So what's happening here?! First, we're telling this TOP that we're going to use some information from a table. Specifically, we're going to use a table whose name is **table1**, and it's location is in the parent network (that's what **../** means). If we were to take our network path **../table1** and write it as a sentence it might read something like this – "hey TouchDesigner I want you to use a table DAT for this TOP, to find it look in the network above me for something named **table1**." Next we have an expression that's telling this TOP what row to pull text from. We just used a similar expression when we were setting our alignment order, and the same idea applies here – me.parent().digits as a sentence might read "hey TouchDesigner, look at my parent (the component that I'm inside of) and find the number at the end of its name." Remember that we named this component **button0**, and that 0 (the digits from that name) correspond to the first row in our table.

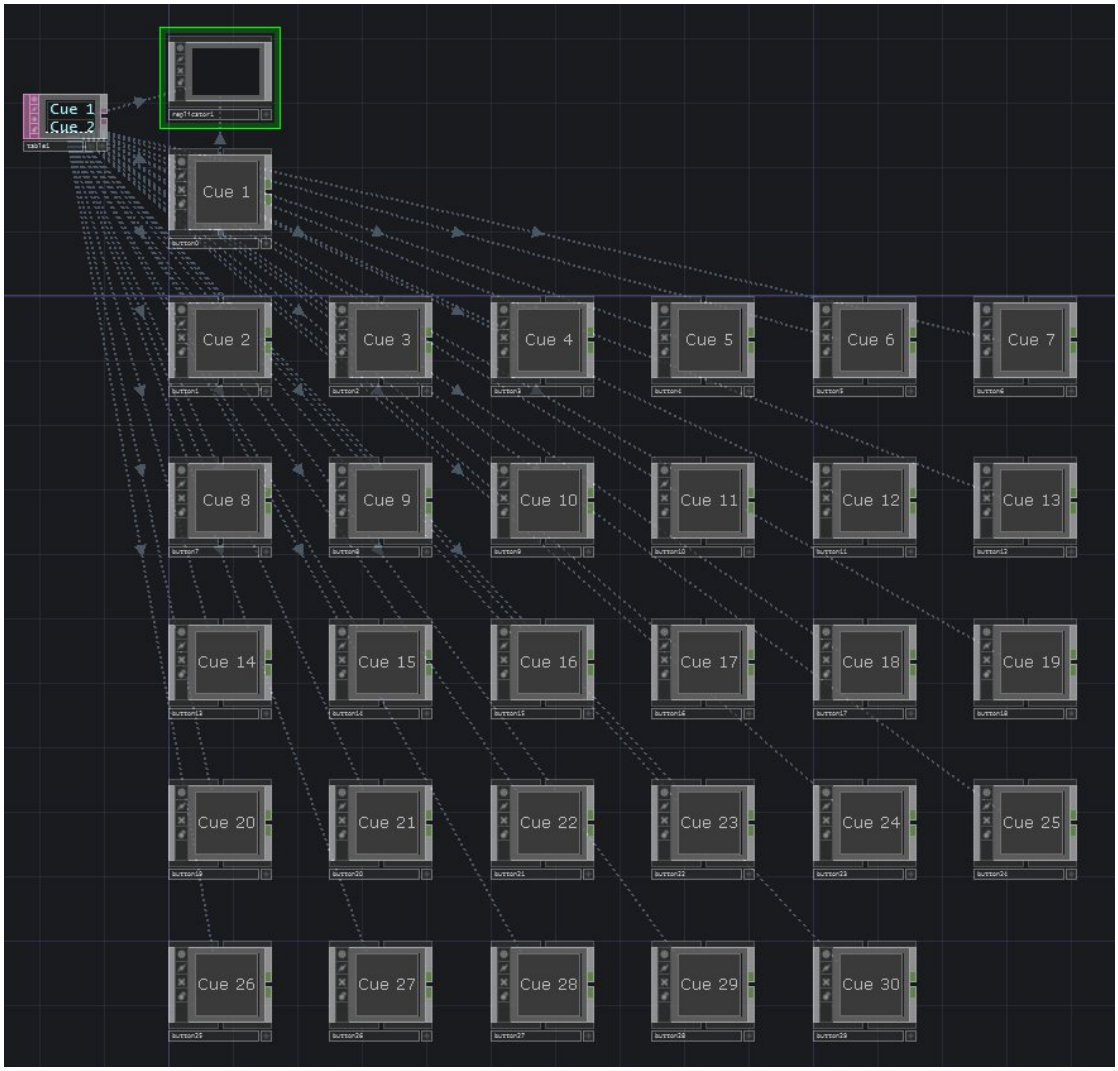
Now it's time to blow things up. Let's add a replicator component to our network.



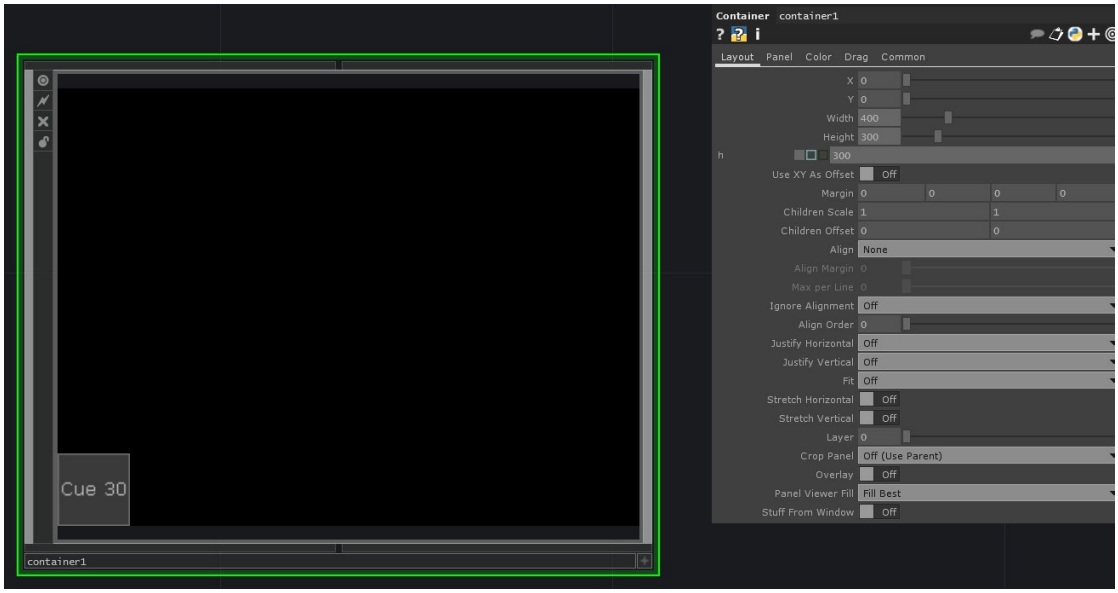
In the parameters for our replicator first specify that **table1** is the template table. Next let's change the Operator Prefix to **button** to match the convention that we already started. Last by not least set your Master Operator to be your **button0**.



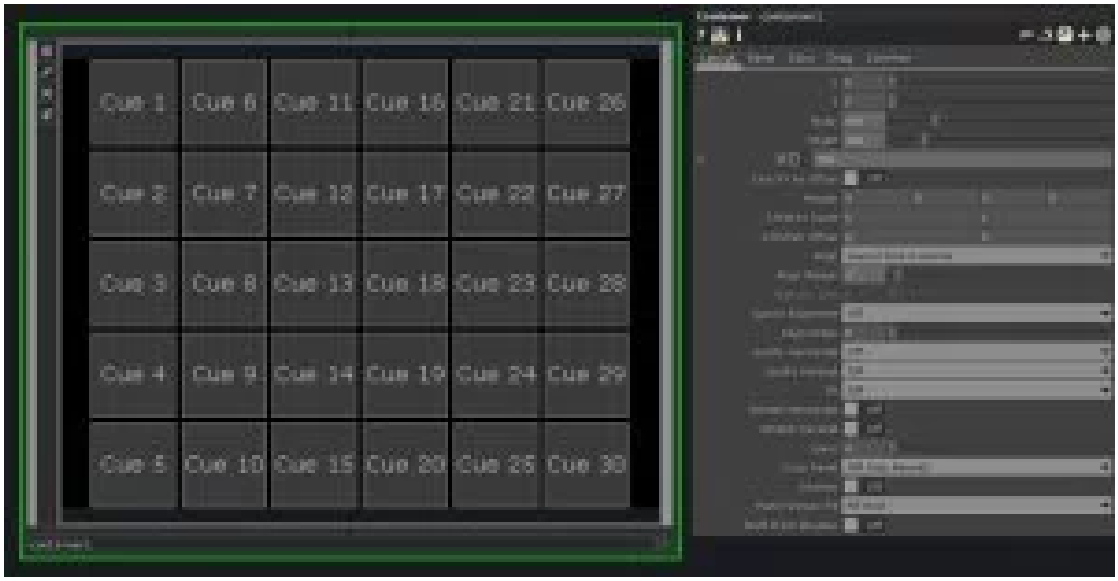
If we set up everything correctly we should now have a total of 30 buttons arranged in a grid in our network. They should also each have a unique name that corresponds to the table that we used to make this array.



Last but not least, let's back up and see what this looks like from the container view. At this point you should be sorely disappointed, because our control panel looks all wrong – in fact it looks like there's only one button. What gives?



The catch here is that we still need to do a little bit of house keeping to get to what we're after. In the Container's parameters on the layout page make sure the Align parameter is set to "Layout Grid Columns" or "Layout Grid Rows" depending on what view better suits your needs. You might also want to play with the Align Margin if you'd like your buttons to have a little breathing room.



Bingo-Bango you've just made some replicator mojo happen. With your new grid of buttons you're ready to do all sorts of fun things. Remember that each of these buttons is based on the template of the master operator that we specified in the replicator. In our case that's **button0**. This means that if you change that operator – maybe you change the color, or add an expression inside of the button – when you click the "recreate all operators" in the replicator, this remakes your buttons with those changes applied to every button.

Happy replicating.

A big thank you to Richard Burns and Space Monkeys for sharing this brief tutorial that inspired me to do some more looking and playing with Replicators:

This entry was posted in How-To, Live Performance, media design, Media Installations, production, Programming, Software, TouchDesigner and tagged Arizona, Arizona State University, ASU, Grad School, graduate school, programming, TouchDesigner, TouchDesigner Tutorial, Wonder Dome on March 6, 2014 [https://matthewragan.com/2014/03/06/touchdesigner-replicators-and-buttons-and-tables-oh-my/] .

One thought on “TouchDesigner | Replicators, and Buttons, and Tables, oh my”

Pingback: Inside Wonder Dome | TouchDesigner | Matthew Ragan

Comments are closed.