DIGIWEAVE

FRIENDSHIP BRACELETS 2.0

A Tutorial by Angela Wang and Catherine Yeh

I. Introduction to DigiWeave

DigiWeave is a programming language that allows users to design friendship bracelet patterns. Perfect for lovers of arts & crafts and computer scientists alike, DigiWeave's goal is to offer an innovative, user-friendly tool that fosters creative expression through technology. DigiWeave strives to be a simple, straightforward language, so that everyone, regardless of coding experience, can design with it. There are currently many programming languages that generate graphics, though most require some knowledge and experience with coding and are not built for non-programmers, and there isn't one centered on friendship bracelet patterns. As such, DigiWeave's simplicity and accessibility provides a tool for everyone to create and visualize friendship bracelet patterns as well as a way to share their creations.

II. The Basics

In DigiWeave, a working program consists of a pattern that represents the pattern of a friendship bracelet.

What is a pattern made of?

A pattern, as defined by DigiWeave, has three required components: a **name**, 1+ **strings**, and 1+ rows of **knots**. Let's break down each of these components.

Name

DigiWeave allows you to name your patterns (ex. `ARROW` or `HEART`). Names can be as specific or vague as you want-- it's completely up to you! However, they can be useful for differentiating patterns later or reminding yourself about what the finished product should look like.

Strings

Physical friendship bracelets are made from embroidery thread/strings of some sort. Depending on how the strings are knotted together, you can produce different patterns.

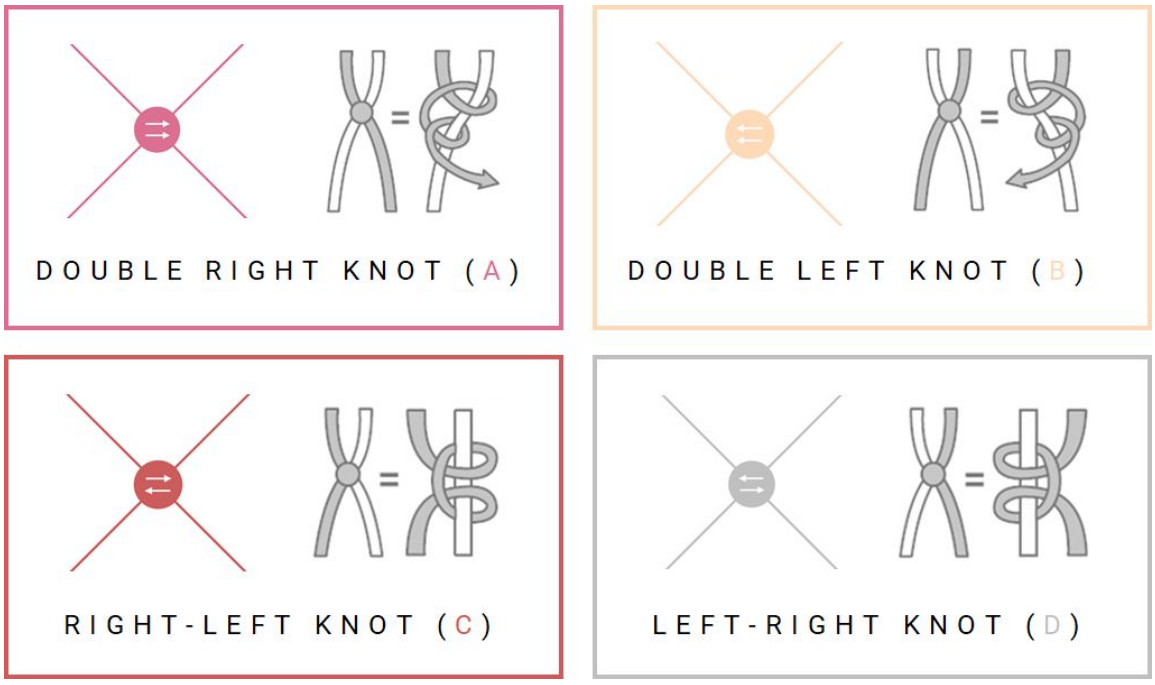
Digiweave allows you to specify a *number* and the *colors* of your strings to more accurately simulate the bracelet making process.

Knots

Knots are the basic building blocks of friendship bracelets. They are created when two neighboring strings are tied together. A sequence of knots creates a row. We will go over the different types of knots in the next section.

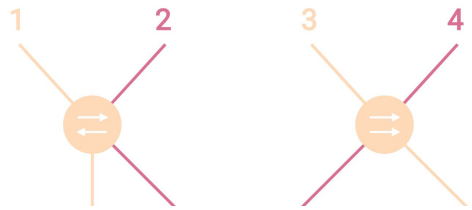
What are the different types of knots?

There are four basic knots you need to know to make a friendship bracelet pattern: a **double right knot**, a **double left knot**, a **right-left knot**, and a **left-right knot**. For simplicity, users will type `A`, `B`, `C`, and `D` in their programs to represent these knots. A diagram and table summarizing these knots is provided below.



	Strings *switch* positions	Strings *don't switch* positions
Left string ends up on top	**Double Right Knot (A)**	**Right-Left Knot (C)**
Right string ends up on top	**Double Left Knot (B)**	**Left-Right Knot (D)**

There is also the **empty knot**, which is represented as `_` (underscore) in DigiWeave. The empty knot is used as a placeholder if there is no knot between two neighboring strings. For example, in the image below, there is a knot between strings 1 & 2 and 3 & 4 but not 2 & 3, so we tell DigiWeave that there is an empty knot between strings 2 & 3.



Thus, you can represent a row of knots by typing any sequence of the characters: `A`, `B`, `C`, `D`, and `_`. For example, to represent the row in the image above, we would simply type: `C_A`.

III. Writing Your First Program

Now that you know the basic concepts needed to make a friendship bracelet pattern, let's write your first program in DigiWeave. This pattern will consist of a single row to begin with (you'll be done in three lines of code!).

Writing a .fbp file

Programs written in DigiWeave are saved as `.fbp` (friendship bracelet pattern) files. To get started, open up a new file in your preferred text editor/IDE and save it as `MyFirstProgram.fbp`.

The name operation

The pattern's name must always come first in your `.fbp` file. Let's name our first pattern, `SINGLEROW`. To do this, use the syntax below:

```
(name SINGLEROW)
```

And that's it! Remember to put a space after the `name` keyword and to surround this expression with parentheses. All keywords (`name`, `strings`, `repeat`) should be typed in lowercase letters.

The strings operation

Next, we have to specify how many strings we want in our pattern and what colors they are. This must come next in your `.fbp` file.

Let's use 6 strings in our pattern, but feel free to choose whatever colors you'd like. Digiweave will accept any of the 140 supported html color names (ex. SALMON) or a hex code (ex. #000 or #73C6B6) as valid input. Make sure to include the # if you choose the latter option. To set up the strings for your pattern, use the syntax below:

```
(name SINGLEROW)

(strings 6 CornflowerBlue CornflowerBlue Coral Coral BurlyWood
BurlyWood)
```

You can either put the `strings` expression on a new line or on the same line as the `name` expression but separated by a space. Make sure that the number of strings (ex. 6) comes directly after the `strings` keyword and that all of your colors are separated by a space or new line character as well. The number of strings must be equal to the number of colors inputted afterward. The `strings` expression must also be surrounded by parentheses.

Note that capitalization for html colors doesn't matter, so it's up to user preference.

Adding a row

Alright, now that you've successfully named your pattern and specified the number & colors of strings, it's time to add a row of knots! Remember that in DigiWeave, the different knots are represented as `A` (double right knot), `B` (double left knot), `C` (right-left knot), `D` (left-right knot), or `_` (empty knot).

In a row, there should be 1 less knot than than the number of strings. So in our case, since we're using 6 strings, our row should have 5 knots. Make sure to adhere to this rule, or else DigiWeave will throw an exception.

Let's add a row of 5 `A` knots (or 5 double right knots). To do so, follow the syntax below.

```
(name SINGLEROW)

(strings 6 CornflowerBlue CornflowerBlue Coral Coral BurlyWood
BurlyWood)

AAAAA
```

Yup, it really is as easy as typing 5 A's (or whatever sequence of `chars` represent the knots you want to make). And once again, it doesn't matter whether you put rows on the

same line separated by a space, or on new lines-- both are equally valid programs. The knots need to be typed in all uppercase letters though.

Hooray! You've officially finished writing your first program in DigiWeave!

Running your .fbp file

Once you've saved your `MyFirstProgram.fbp` file, open up a terminal window to run your program. Navigate to the **code/lang** directory, and from there, you can use the *dotnet run* command like so:

```
> dotnet run MyFirstProgram.fbp
```

If your `.fbp` file is located in another folder, make sure to include the complete file path on the command line.

DigiWeave produces two kinds of outputs from your code that represent your created pattern: 1) **evaluator output** and 2) **SVG output**.

Evaluator output

After your program runs, DigiWeave will print your pattern to the terminal to show the results from the evaluator. Here, you can see your pattern's name, and all the rows of knots that make up your pattern to check whether your program was interpreted correctly. For example, you should see something like this after running your `MyFirstProgram.fbp` file:

```
Your pattern has been saved to ...\output\SINGLEROW.html

"Pattern Name: SINGLEROW

Row
1   CornflowerBlue >> CornflowerBlue >> CornflowerBlue >>
    CornflowerBlue >> CornflowerBlue >> "
```

Note that the evaluator prints double right knots as >>, double left knots as <<, right-left knots as >, left-right knots as <, and empty knots as _. For non-empty knots, the color of the knot is printed to the left of the symbol that represents it.

SVG output

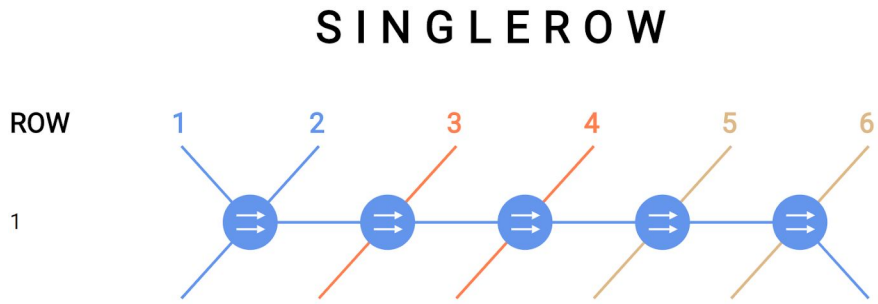
Your pattern will also be saved as an SVG embedded inside an html file. This visual representation of your pattern also shows the sequence of knots and movement of

strings so you can recreate it in real life! Refer back to Section II if you need a reminder of how to create each type of knot or how to translate the symbols used in the SVG diagrams into knot types.

We will go over how to access these SVGs in the next section, but note that the location of this file is also printed to the terminal (see the "`Your pattern has been saved to…`" message above).

Accessing your pattern as an SVG

To view your pattern, first navigate to the **output** folder. Your pattern should be saved as an html file with the same name as what you specified. For example, the program you just wrote should appear as `SINGLEROW.html`. Simply open this file to view your pattern in a browser. Your SVG should look something like this:



And sure enough, we have a row with five double right knots! Huzzah!

IV. More Advanced Patterns

A single row is cool, but let's be honest, that hardly makes for a very interesting friendship bracelet pattern. So, how can you take DigiWeave to the next level?

The repeat operation

To make more advanced patterns, it is extremely useful to be able to repeat rows. So now let's try extending `MyFirstProgram.fbp` to include the repeat operation. Since we're working with 6 strings, why don't we try repeating our row 6 times? To do so, use the following syntax:

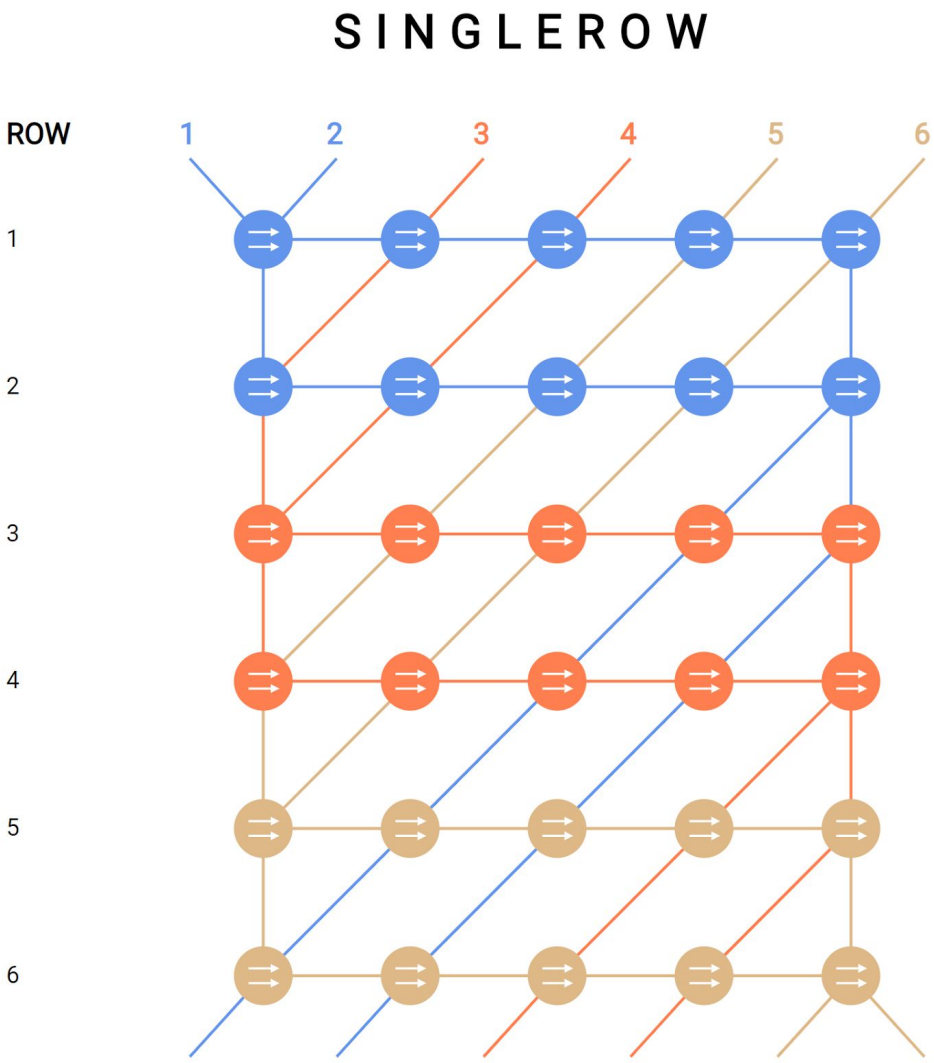
```
(name SINGLEROW)

(strings 6 CornflowerBlue CornflowerBlue Coral Coral BurlyWood
BurlyWood)
```

```
(repeat 6 AAAAA)
```

Note how the `repeat` keyword is also followed first by the number of times to repeat and then the rows to repeat. This expression should also be surrounded by parentheses and again, can be placed on the same line separated by a space from the other code or a new line. You can also nest `repeat` operations to create even more advanced programs (ex. `(repeat 2 AAAA (repeat 3 ABAB))`).

After updating your code and running `MyFirstProgram.fbp` again, your SVG should look like this:



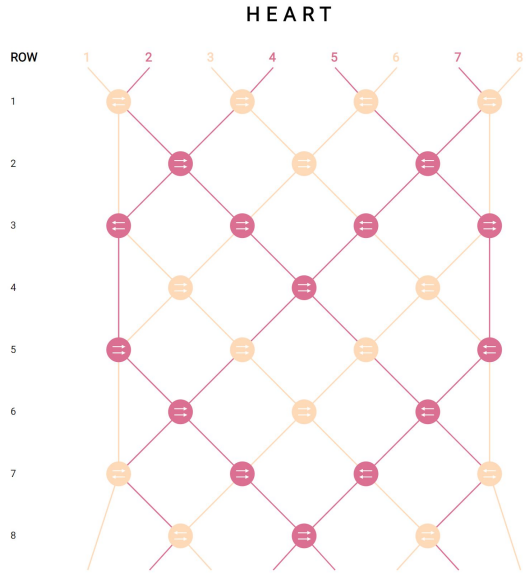
Now that's pretty awesome, right?

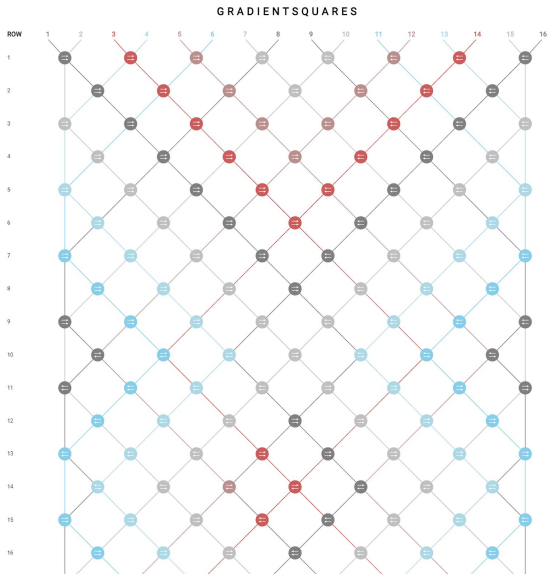
Other examples

The DigiWeave repository also includes a few premade examples to inspire your own digital bracelet making. All of the code for these examples is located in the **examples** folder. To run an example, navigate inside the **code/lang** folder and run the *dotnet run* command on the command line. For example:

```
> dotnet run ../../examples/example-1.fbp
```

Will run and produce output for example 1 (which happens to be the same pattern you just made!). All the output should be saved to the **output** folder, and you can view the SVGs in your browser in the same way as described earlier. Here is a sneak peek of what you can expect:





For an additional exercise, try extending the example programs by adding more rows, repeats, or changing the types/orders of knots to see what happens.

And that's pretty much all you need to know! Enjoy making your own friendship bracelet patterns-- the possibilities are truly endless :).