Drag-and-Drop Order

Alice's drag-and-drop interface is nice, but sometimes it takes a little bit of getting used to for those who are used to typing code from left to right. This is a brief tutorial on the order in which one constructs Alice expressions. In the interest of those who are not familiar with programming, I have not used parameters.

Tutorial

Let's suppose that I want to construct a function that checks whether two objects in Alice are overlapping (that is, overlapping). This will be a world-level function that (eventually) takes two objects as parameters. Start by creating a new Alice world, and add two actual objects to the world. You will need those objects when you construct this function, although, once you're done constructing it, you can throw them away. Here are my two objects:

If you want to follow with this particular setup, you're welcome to download this Alice world [here].

Now, go to the world's functions and create a new function. I'm calling it overlaps. Since we're asking the question of whether or not two objects are overlapping, the function returns a boolean.
Let's begin by dragging the if-statement template over the "doNothing." This is the outermost portion of the expression.

The template requires a condition, so just pick true or false. There's not much point in picking an expression because we're going to replace it anyways. The purpose of the conditional is to check whether or not the distance between the object parameters is small enough to consider them to be overlapping. In other words, is the distance less than necessary for a normal person to say they are overlapping?

We have to build the statement by using a general template. Under the "Math" tab of the world's functions, drag the a < b template over the "true" in the If-statement.

At the prompt following the drop, just pick any numbers for a and b. These, too, will be replaced. I've picked one and two.
Now, click on one of the objects in the object tree, and click on the functions tab. Drag its "distance to" function over the left side of the inequality, and select bunny -> the entire bunny.

So far, we've completed half of the conditional. On the right-hand of the inequality, we will use a simplification and assume both that the two objects are on the same plane (both on the ground) and that the width and depth are essentially equal. Then, the distance between the centers of the two objects is just half the width of one + half the width of the other.
Go to the boar's functions and drag boar's width over the 2.

Then, click the arrow to the right of the word "width" and select math → boar's width / → 2.

Notice the parenthesis Alice has put in.
Now, click on the arrow outside the right parenthesis, and choose math, addition, and select a number (to be replaced).

Now, we're going to replace that number with the same expression. We just have to be careful with which arrows we're clicking on so that we know exactly what order we're evaluating in. See the screenshot below and watch where the arrows are clicked.
Finally, we want the function to return true if the condition is satisfied, so drag the return under the 'if' and select true.

Also, we want to return false if that is not the case, so drag another return under the else, and select false.
For good measure, also change the return statement at the bottom of the function to false. (Note: This is not strictly necessary, but it makes it less likely that someone will misread your code.)

You can download the world at this point [here](#).

Now, we can immediately test our function by writing some code that depends on it in world.my first method. Start by dragging an "if statement" into this method, as below:

Then, replace true with our new constructed function. (I hope everyone realizes by now that Alice doesn't let you build an expression from left to right):
Then, we want some visible indication from Alice of whether or not the two objects are overlapping. Let us then have the boar speak to the matter. Click on the board, and drag his "say" template under the If, and select other.

I'm having him say "We are overlapping. How can this be?" You might choose whatever you like. When you are done with this step, your screen should look like this, although probably with different text.
Now, do the same thing for the "Else" part of the If statement. When you are done, your first method should look something like this:

```
world.my_first_method
```

```
boar say We are overlapping, how can this be? more...
```

```
boar say Checking... We are not overlapping. more...
```

At this point, you can play your world, and notice that the boar will indicate the two characters are not overlapping. Then, stop the world and drag the boar on top of the rabbit, or vice versa, they are clearly overlapping, and Play the world again:

Note that you might have to bring the two rather close to get the desired effect, because our function was very crude, since it made the assumption (for simplicity) that the width, height, and depth are all the same.

In this tutorial, we talked about how to construct an Alice expression by dragging and dropping a template, selecting some arbitrary values, and then replacing those arbitrary values with meaningful expressions. The point is that Alice expressions have to be constructed, for the most part, from the outside in, and that expressions containing arithmetic have to be done one step at a time. This concludes the tutorial on Drag-and-Drop. When you learn about parameters, you will be able to turn the function we created into a truly useful one that can check for overlap between any two objects in your world.