DragginMath uses colors to help you understand what you see and what to do.

Colors show how things are related to each other in a math expression's structure. When the parts of an operator tree are enclosed in a rectangle with a colored background, that means those parts are tied together by the operator at the top of that rectangle. These rectangles stack on top of each other, each a slightly different color. Colors are selected at random. They have no meaning other than to show you how the parts group together.

When dragging math around on the screen, colors around your fingertip tell you which *mode* you are in. The mode determines what you are able to accomplish when you drag: different modes do different things.

These two uses of color are not related to each other. For example, when DragginMath paints a blue background in some rectangle on the screen, that has nothing to do with Blue Mode.

What You Need To Remember

Drag Up for Blue Mode

- Associate
- Distribute
- Factor
- Invert (Solve)
- Replicate
- Simplify
- Substitute
- Superpose

Drag Sideways for Purple Mode
- Commute
- Cancel

Drag Down for Green Mode
- Evaluate
- Expand

Flick or Double-Tap in Red Mode
- Convert Unary Operators to Binary
- Convert Signs
- Factor Numbers
- Convert to Linear Notation

What These Words Mean

When you touch a DragginMath tree node (a number, variable, or operator), a red circle appears around your fingertip. You are now in Red Mode. This means DragginMath knows you intend to do *something* with that node, but you haven't done it yet. What you *can* do with it depends on what you do next.

If you drag your fingertip *up* a short distance, the red circle turns blue. This means you are now in Blue Mode, which allows you to do certain things that will be described to you soon. Once the circle turns blue, it never changes back to red, or to any other color. If you decide at this point that you don't want to do Blue Mode actions, lift your fingertip: the circle disappears, and the tree node goes back where it came from.

If you drag your fingertip *directly sideways* so it moves *past* some other tree node, the red circle turns purple. This means you are now in Purple Mode, which allows you to do certain things that will be described to you soon. Once the circle turns purple, it never changes back to red, or to any other color. If you decide at this point that you don't want to do Purple Mode actions, lift your fingertip: the circle disappears, and the tree node goes back where it came from.

If you drag your fingertip *down* a short distance, the red circle turns green. This means you are now in Green Mode, which allows you to do certain things that will be described to you soon. Once the circle turns green, it never changes back to red, or to any other color. If you decide at this point that you don't want to do Green Mode actions, lift your fingertip: the circle disappears, and the tree node goes back where it came from.

What can you do in Purple Mode? There are two things.

You can **commute** the operands of a binary operator. If your expression is **2+3**, drag the **2** sideways *past* the **3**, or drag the **3** sideways *past* the **2**. Either way, the tree redraws, and the expression at the top of the screen now says **3+2**. It *looks* different, but it *means* the same thing. This also works for multiplication: **2*3** becomes **3*2**.

+ and * are commutative, so you expect this. The other operators are not commutative. But it is possible to commute some other operators if we allow DragginMath to *do other things also.* For example, commuting **3–2** results in **¯2+3**. It *looks* different, but it *means* the same thing. So some *non-commutative* operators are still *commutable*. Which? Try them and find out.

You can **cancel** certain operands. For example, if your expression is **3–3**, dragging either **3** sideways past the other causes that part of the tree to change into **0**. This also works for division: **3÷3** changes into **1**. Cancellation works for things much more complicated than simple numbers. All that matters is that both operands of either – or ÷ are equal. DragginMath can *sometimes* figure out that complicated operands are equal, even when that is not obvious at first glance. For example, **(ab+3)–(3+ba)** will cancel without help from you, but **(a+b+c)-(a+(b+c))** will not.

What can you do in Green Mode? There are two things.

You can **evaluate** expressions. If your expression is **2+3**, drag the **2** *down* to enter Green Mode, then drop the **2** on the +. When your finger is directly over the +, a black border appears around the **2+3** tree, the background flashes into a different color, and a small target icon appears in the upper left corner of the screen. Also, you will hear a click. This means you are *on target*. If your finger drifts off the +, the black border disappears, the background becomes white again, the target icon goes away, and you will hear a click at a lower pitch. This means you are *off target*. Drag the **2** down, then onto the +, see that you are *on* target, and lift your fingertip. The result is **5**. It *looks* different, but it *means* the same thing.

DragginMath can evaluate any expression in terms of *integer arithmetic*. We did that here with **2+3**, which evaluates to **5**. We can also evaluate **12÷4**, which becomes **3**. What about **4÷12**? That becomes **1÷3**, the equivalent fraction in lowest terms. Why is it not 0.333333…? Because DragginMath is a tool for algebra, not for calculation. In the context of an algebra class, **1÷3** *is the correct result*. Similarly, DragginMath has no way to tell you that **√50** ≈ 7.071067… Instead, it tells you that **√50** = **√(25*2)** = **√25*√2** = **5∗√2**. In the context of an algebra class, **5∗√2** *is the correct result*.

Expressions containing free variables cannot evaluate to a number. But they sometimes **expand** into another expression, which may be simpler or more complicated. Depending on what you are trying to do, the result may be more useful. For example, enter **(a+b)↑2**. Drag the **2** down into Green Mode, then onto ↑ and see what happens. Yes, this result is correct. By the way: you can arrive at this same result using a combination of Blue Mode actions. It is good to understand the relationship between the Green Mode expansion and the equivalent Blue Mode actions.

If DragginMath does not evaluate an expression fully, that might mean no one else can either, at least not in its current form. To evaluate further, DragginMath might need your help to change the expression's structure in some way before proceeding. Or it may be that you and DragginMath have already done all that can be done.

DragginMath evaluates or expands the smallest common subtree of the dragged node and its target. That might sound complicated, but it isn't. The easiest move to *explain* is to drag any *leaf node* within the tree you want to evaluate or expand down into Green Mode, then up onto the root of that tree. This always works. The easiest move to *make* is often different and shorter, but harder to explain in a document like this. Green Mode even allows you to *exclude* certain parts of the operator tree from evaluation. We won't go into those details here. However you go about evaluating, the most important thing is to first drag something *down* to enter Green Mode.

What can you do in Blue Mode? This is a busy place. You can do many things here. To enter Blue Mode, drag something *up*. Then drag it *onto* something else to make things happen, just as you did in Green Mode. You will see and hear the same cues whenever you are *on target*.

You can **associate**. If you have **(2+3)+4**, drag the + connected to **2** and **3** onto the + connected to **4**. See the operator tree become **2+(3+4)**. It *looks* different, but it *means* the same thing. This also works for multiplication: **(2∗3)∗4** becomes **2∗(3∗4)**.

+ and ∗ are associative, so you expect this. The other operators are not associative. But it is possible to reassociate *some* other operators if we allow DragginMath to *do other things also*. For example, reassociating **(4–3)–2** results in **4–(3+2)**. It *looks* different, but it *means* the same thing. So some *non-associative* operators are still *associable*. Which? Try them and find out.

You can **distribute**. If you have **2∗(3+4)**, drag + up onto ∗ and see the operator tree become **2∗3+2∗4**. This also works for much more complicated distributions of ∗ and ÷ over + and –. For example, you can distribute **ab(c+d–e)÷f** by dragging − up over ÷. Distribution also works for raise↑, root√, and log↓ over other operators in specific combinations having to do with the rules of exponents.

You can **factor**. If you have **2+2+2+2**, drag the first **2** up onto the last **2** and see the operator tree become **4∗2** (*factoring by counting*). This also works for ∗. If you have **2∗3+2∗4**, drag either **2** up onto the other **2** and see **2∗(3+4)** (*factoring by extraction*). Notice this reverses distribution. You can also factor expressions involving ↑ √ ↓.

You can **invert**. This is how you solve equations. Drag a tree node up onto a relation such as = to move that node to the other side of the relation. For example, if you have **x+3=7**, drag **3** up onto = and see **x=7−3**. If you try to move all of one side of a relation, a dialog asks if you want to do this using subtraction or division.

You can **replicate**. If you have **3∗4**, drag the **3** onto the ∗. See the operator tree become **2*4+4**. Then drag the **2** onto its neighboring ∗ and see **4+4+4**. Or, starting over with **3∗4**, drag the **4** onto the ∗. See the operator tree become **3+3*3**. This also works for ↑. Note that replicating **x∗0** results in **0**, and replicating **x∗1** results in **x**. Replicating **x↑0** results in **1**, and replicating **x↑1** results in **x**. Replicating on addition results in a sum of **1**s. This is actually useful.

You can **simplify**. If you have **0+a**, drag **0** up onto + and see the operator tree become just **a**. If you have **-7**, drag **7** up onto **-** and see ¯7. If you have **- -7**, drag the **7** up onto the first **-** and see **7**. If you have **-⅟ 7**, drag **⅟** up onto **-** and see **⅟ -7**. Many trivial simplifications and rearrangements of signs work this way. If it seems that it should be possible, it probably is.

You can **substitute**. To do this, you need at least one actual equation and some other expression on the screen. If you have **a=b+c; 2a–a÷3**, drag **a** in the equation onto –. See the operator tree become **2∗(b+c)–(b+c)÷3**. Now drag **b+c** onto –. See the operator tree become **2a–a÷3** again. The source of your substitution must be one side of a true equation. Inequalities won't work as a substitution source, but any expression or subexpression works as a target. A target does not have to be a root, but it does have to be in an expression separate from the source. For convenience, this also works dragging down into Green Mode.

You can **superpose** an equation onto another relation. If you have **x+y=5; x−y=3**, you can superpose by dragging either = up onto the other. This adds the the two relations. If you drag the first =, see the result **x+y=5; x−y+x+y=3+5**. The drag source must be an = ; the target can be any relation. For convenience, this also works dragging down into Green Mode.

What can you do in Red Mode? These actions are either *flicks* or *double-taps*, not *drags*. To flick, touch the tree node, then flick your finger off the surface. This may require a little practice to do it reliably. To double-tap, tap the node twice, quickly.

If you have **-x**, flicking down on **-** converts it to **0–x**. This works for most unary operators, converting them into their equivalent binary operators and operands. If you have the number **¯3**, where the minus is the sign of the number, not the negate (unary minus) operator, flicking down on **¯3** converts it into a negate operator on the positive number **3**.

If you flick left on the root of an operator tree, DragginMath asks if you want to discard that expression. If there is only one expression on the screen, you cannot discard it.

If you flick up on an operator in the tree, its subtree compresses into a line of symbols. Flick down on the line of symbols to expand it back into a tree.

If you double-tap a number, it is replaced by a tree of its factors. For example, if you double-tap **4567890**, it becomes **2∗3∗5∗43∗3541**.

If you double-tap ±, the whole expression is replaced by two copies of itself, where one copy contains + at the same location as ±, and the other copy contains –. Double-tapping is the only way to evaluate ±. Green Mode has no effect on it.

The Trick

Perhaps this feels like a lot to remember. The easiest way might be this: Purple Mode does Commute and Cancel, Green Mode does Evaluate and Expand, Red Mode is about a flicking and double-tapping, and Blue Mode does Everything Else. Experience with actual users like yourself shows this quickly becomes second nature after only a little practice.