DragginMath: About Colors and Modes

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 (Anti-Distribute)
- Invert (Solve)
- Replicate
- Simplify
- Substitute

Superpose

Drag Sideways for Purple Mode
- Commute
- Cancel

Drag Down for Green Mode
- Nothing going on here in this version

Flick or Double-Tap in Red Mode
- Convert Operators
- Convert Signs
- Copy
- Delete
- Evaluate
- Expand
- Factor Numbers
- Fold / Unfold
- Inject
- Replace
- Split

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 simply lift your fingertip while in Red Mode, nothing happens, and the tree node goes back where it came from.

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 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. You will see and hear several cues when you are *on target*: an audible click, the screen background changes color, a black border appears around the drop target, and a target icon appears in the upper left corner.

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↓ with 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, where the associated operator is inverted. For example, if you have **x+3=7**, drag **3** up onto = to obtain **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 the ∗ 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. Also, the combinatorial operators **! ¡** ℗ © replicate in interesting ways that help you understand what they mean.

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. It doesn't work in Red or Purple Mode, which is a common error.

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 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 = sign; the target can be any relation. For convenience, this also works dragging down into Green Mode. It doesn't work in Red or Purple Mode, which is a common error.

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 according to the technical definition of that word. 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? At this time, *nothing*.

Evaluate and **Expand** used to be the focus of Green Mode, and existing users may still be accustomed to this. These have been simplified to double-tapping and moved to Red Mode. New features will eventually appear under GreenMode, so don't forget that Green Mode is here.

To enter Green Mode, drag *down* until things turn green.

What can you do in Red Mode? You won't actually see anything turn red for these actions because they are either *flicks* or *double-taps*, not *drags*. The point is: you haven’t entered Blue, Green, or Purple Mode. To flick, quickly brush your finger over the surface of the node, where the direction of your flick matters. This may require a little practice to do it reliably. To double-tap, tap the node twice, quickly.

You can **convert some binary operators** into equivalent forms. For example, flick *down* on the root of **a−b** to obtain **-(b−a)**. These conversions are occasionally actually necessary. Or flick down on **aⓅb** to obtain **a!÷(a-b)!**. *Beware*: this is easy to do by accident.

You can **convert some unary operators** into their equivalent binary operators. For example, if you have **-x**, flicking *down* on **-** converts it to **0–x**.

You can **convert a signed number** into an operator on an unsigned number. Consider 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 **-3**, which is a negate operator on the positive number **3**.

You can **copy** any node by flicking *left* on it. A dialog appears, asking if you want to copy the node *as a node* or *as text*. If you copy *as a node*, it is appended to the set of trees already on the screen. If you copy *as text*, you can paste it into other apps.

You can **delete** an operator tree by flicking *left* on its root. A dialog appears, asking if you want to delete or copy. If there is only one expression on the screen, you cannot delete it.

You can **evaluate** expressions. If your expression is **2+3**, double-tap + to obtain **5**. If your expression is more complicated, simply double-tap the root of whatever subtree you want to evaluate. For example, if have **2(3+4−5)** and you want to evaluate only **3+4−5**, double-tap − to obtain **2*2**.

DragginMath evaluates expressions using *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 variables cannot evaluate to a number. But they sometimes **expand** into other expressions, which may be simpler or more complicated. Depending on what you are trying to do, the expanded result may be more useful. For example, enter **(a+b)↑2**, then double-tap ↑ 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 this expansion and the equivalent Blue Mode actions.

If DragginMath does not evaluate or expand an expression fully, that might mean no one else can either, at least not in its current form. To go 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.

You can **factor a number** by double-tapping it. For example, if you double-tap **4567890**, it becomes **2∗3∗5∗43∗3541**. If you then double-tap the ∗ at the top of the tree, it evaluates back to **4567890**.

You can **fold** all or any part of an operator tree into DragginMath's linear notation by flicking *up* on an operator node. Flick *down* on the resulting line of symbols to **unfold** it into a tree again. Folding or unfolding doesn't affect the meaning of an expression in any way other than its appearance, and folding/unfolding is not remembered in History. Many interactions with a folded tree cause it to unfold automatically. You cannot interact with elements *inside* a folded tree, only its root.

You can **inject** terms into an existing expression in a ways that are algebraically safe. Flick *right* on a node, then see a dialog asking what you want to inject. Enter an expression using the keyboard (remember to tap ↲ when you are finished), then select *how and where* you want to inject it around the node you flicked. For example, if your original expression is simply **x**, flick *right* on the **x** node, enter **1** in the dialog, then tap the **□+■−□** button to obtain **1+x−1**. The exact behavior depends on the kind of node you flick and which button you tap in the dialog. The design goal is to make it easy to do what you probably want in that situation, but you may have to manipulate the result farther to suit your exact need. This feature has its own ℹ️ page as **Balanced Injection**.

You can associate an expression with a simple name using the =: operator, where the expression is on the left, and the name is an uppercase letter on the right. As usual with DragginMath, these expressions can range from simple values to things much more complicated. For any Named Expression with an active definition, a corresponding lowercase variable (e.g. **a** for **A**) can **replace** itself in an expression with the associated value simply by double-tapping. You must do this deliberately; it doesn't happen automatically when you evaluate an expression. If there is no associated definition, nothing happens when you double-tap. Variable nodes with definitions appear with a thicker border that is easy to recognize.

You can **split** an expression containing ± by double-tapping the ± operator. 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 move forward with ±.

The Trick

Perhaps this feels like a lot to remember. The easiest way might be this: Purple Mode does Commute and Cancel, Red Mode is about a flicking and double-tapping, Green Mode currently does nothing, and Blue Mode does Everything Else. Experience with actual users like yourself shows this quickly becomes second nature after only a little practice.