# Assignment IV: EmojiArt

## Objective

The goal of this assignment is to learn about how to deal with multi-touch gestures.

Be sure to review the Hints section below!

Also, check out the latest in the Evaluation section to make sure you understand what you are going to be evaluated on with this assignment.

## Due

This assignment is due before you watch Lecture 9.

## Materials

- Download the version of EmojiArt from Lecture 8 and start working on this assignment from there.

## Required Tasks

1. Download the version of EmojiArt from Lecture 8. Do not break anything that is working there as part of your solution to this assignment.
2. Support the selection of one or more of the emojis which have been dragged into your EmojiArt document (i.e. you're selecting the emojis in the document, *not* the ones in the palette at the top). You can show which emojis are selected in any way you'd like. The selection is not persistent (in other words, restarting your app will not preserve the selection).
3. Tapping on an unselected emoji should select it.
4. Tapping on a selected emoji should unselect it.
5. Single-tapping on the background of your EmojiArt (i.e. single-tapping anywhere except on an emoji) should deselect all emoji.
6. Dragging a selected emoji should move the entire selection to follow the user's finger.
7. If the user makes a dragging gesture when there is no selection, pan the entire document.
8. If the user makes a pinching gesture anywhere in the EmojiArt document and there is a selection, all of the emojis in the selection should be scaled by the amount of the pinch.
9. If there is no selection at the time of a pinch, the entire document should be scaled.
10. Make it possible to delete emojis from the EmojiArt document. This Required Task is intentionally not saying what user-interface actions should cause this. Be creative and try to find a way to delete the emojis that feels comfortable and intuitive.

## Hints

1. Note that your EmojiArt now supports both single tap and double tap gestures on the background. You will likely want to use the `Gesture` function `.exclusively(before:)` to make sure double taps don't get missed because SwiftUI recognizes the single tap part of a double tap and doesn't give an opportunity for the second tap to be recognized as part of a double tap gesture.
2. Your selection is not part of your Model. It is purely a way of letting the user express which emoji they want to resize or move. Thus you will want to maintain your selection in your UI code somewhere.
3. A `Set` might be a good data structure to store the set of selected emoji since there's no "order" to the selection (like an `Array` would imply). However, since you're likely managing your selection based on the `Emoji`s' `Identifiable`-ness anyway, it probably doesn't matter. We marked `EmojiArt.Emoji` as `Hashable` just in case you want to put it in a `Set`.
4. If you do choose to use a `Set` for your selection, adding a `toggleMatching` function via an `extension` (that adds/removes an element to/from the `Set` based on whether it's already there based on `Identifiable`) might be nice.
5. Note that there is a difference between the `contains(matching:)` function we added to `Collection` in `EmojiArtExtensions` and the `contains()` function that is built into the Swift's `Set`. The later `contains` knows nothing about `Identifiable`. So if you decide to use a `Set<EmojiArt.Emoji>` anywhere in your code, you probably want the former.
6. When it comes to pinching, all pinches are always recognized on the entire document (even though what happens on such a pinch depends on whether there's a selection at the time). So you can likely piggy-back your selection resizing on the existing `MagnificationGesture` in EmojiArt.
7. You will not be able to piggy-back your emoji-dragging onto the whole-document panning drag gesture. A drag gesture that starts on a selected emoji does a very different thing than a drag gesture that does not. So a new `DragGesture` will be required to support moving the selection around.
8. Gestures can sometimes interfere with each other (after all, they are all going after the same touch events). For example, if we swap the order of our pan and zoom gestures, they will not work properly. Check out the Gesture documentation for functions like `simultaneously` and `exclusively` which you can use to get your gestures to play together better.
9. As always, we try to make "conditionality" in our SwiftUI `View`s be embedded into the *arguments* to `View` modifiers rather than by using `if-else` statements (whether inside or outside `@ViewBuilder` constructs). This often takes the form of a ternary operator

( `? :` ) but may involve calling a `function` or getting the value of a computed `var` as part of the conditional decision-making.

## Things to Learn

Here is a partial list of concepts this assignment is intended to let you gain practice with or otherwise demonstrate your knowledge of.

1. Gestures

## Evaluation

In all of the assignments this quarter, writing quality code that builds without warnings or errors, and then testing the resulting application and iterating until it functions properly is the goal.

Here are the most common reasons assignments are marked down:

- Project does not build.
- One or more items in the Required Tasks section was not satisfied.
- A fundamental concept was not understood.
- Project does not build without warnings.
- Code is visually sloppy and hard to read (e.g. indentation is not consistent, etc.).
- Your solution is difficult (or impossible) for someone reading the code to understand due to lack of comments, poor variable/method names, poor solution structure, long methods, etc.

Often students ask "how much commenting of my code do I need to do?" The answer is that your code must be easily and completely understandable by anyone reading it.

## Extra Credit

We try to make Extra Credit be opportunities to expand on what you've learned this week. Attempting at least some of these each week is highly recommended to get the most out of this course. How much Extra Credit you earn depends on the scope of the item in question.

If you choose to tackle an Extra Credit item, mark it in your code with comments so your grader can find it.

1. Allow dragging *unselected* emoji separately. In other words, if the user drags an emoji that is part of the selection, move the entire selection (as required above). But if the user drags an emoji that is not part of the selection, then move only that emoji (and do not add it to the selection). You will find that this is a much more comfortable interface for placing emojis.