Assignment IV: Animated Set

Objective

The goal of this assignment is to understand the animation mechanisms described in lecture this week by adding animation to your Set assignment from last week.

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

You have one week to work on this assignment.

Materials

- You will need your code from last week's Set assignment.

Required Tasks

1. ~~Your assignment this week must still play a solo game of Set.~~
2. ~~In this version, though, when there is a match showing and the user chooses another card, do **not** replace the matched cards; instead, discard them (leaving fewer cards in the game).~~
3. Add a "deck" and a "discard pile" to your UI. They can be any size you want and you can put them anywhere you want on screen, but they should not be part of your main grid of cards and they should each look like a stack of cards (for example, they should have the same aspect ratio as the cards that are in play).
4. The deck should contain all the not-yet-dealt cards in the game. They should be "face down" (i.e. you should not be able to see the symbols on them).
5. The discard pile should contain all the cards that have been discarded from the game (i.e. the cards that were discarded because they matched). These cards should be face up (i.e. you *should* be able to see the symbols on the last discarded card). Obviously the discard pile is empty when your game starts.
6. Any time matched cards are discarded, they should be animated to "fly" to the discard pile.
7. ~~You don't need your "Deal 3 More Cards" button any more. Instead, *tapping on the deck* should deal 3 more cards.~~
8. Whenever more cards are dealt into the game for any reason (including to start the game), their appearance should be animated by "flying them" from the deck into place.
9. Note that dealing 3 more cards when a match is showing on the board still should replace those cards and that those matched cards would be flying to the discard pile at the same time as the 3 new cards are flying from the deck (see Extra Credit too).
10. All the card repositioning and resizing that was required by Required Task 2 in last week's assignment must now be animated. If your cards from last week never changed their size or position as cards were dealt or discarded, then fix that this week so that they do.
11. ~~When a match occurs, use some animation (your choice) to draw attention to the match.~~
12. ~~When a mismatch occurs, use some animation (your choice) to draw attention to the mismatch. This animation must be very noticeably different from the animation used to show a match (obviously).~~

Hints

1. The animations for match and mismatch do not necessarily have to repeat forever like we did in Memorize (though that could be cool depending on the animation) and it does not necessarily have to involve the symbols (though it's very easy to imagine the symbols being involved in these animations).
2. Your deck and discard pile have to look like a stack of cards. It's okay if they look like an extremely neatly stacked pile of cards (like Memorize's stack of undealt cards did).
3. Animations that repeat forever can sometimes be problematic when combined with other animations (like the animation that happens when your device rotates, for example). We won't hold it against you if there are some "corner cases" (i.e. rare situations that *don't arise during normal usage*) where your animation gets a little funky (but clicking to advance the game should cause any such problems to disappear).
4. The "back" of a card can look like whatever you want (except that it can't give away what's on the front of the card, of course!).
5. There is no requirement for cards to "flip over". See Extra Credit.
6. None of the Required Tasks require you to create your own `ViewModifier`, but you'd probably learn a lot by doing so if you can fit it into your solution.
7. None of the Required Tasks require using an animation variable like `animatableBonusRemaining` from lecture, but, again, you are more than welcome to do such an animation as part of your solution if you want to challenge yourself a bit.
8. Since you'll be using `matchedGeometryEffect` to transition your cards in and out of existence (as they pass from deck to playing field to discard pile), you probably won't need to use `.transition` for that but depending on how you animate your matched and unmatched cards, it might be something you would want to use there.

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. Animation

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. You can assume that the reader knows the iOS API and knows how the Memorize game code from the lectures works, but should not assume that they already know your (or any) solution to the assignment.

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.

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

1. Have your deck and/or discard pile be either "sloppy" (i.e. not a perfectly neat stack) or show the first few cards slightly offset (so that it looks more like a stack).
2. When you deal 3 more cards and there is a match showing, start animating the matched cards flying to the discard pile *before* the animation of the 3 new cards flying in from the deck starts. In other words, give the user a better impression of "I just replaced these 3 cards for you" by *delaying* the dealing animation a short bit in this scenario. The animations can still overlap, but delaying the dealing one just a little bit can result in a pleasing effect.
3. Make the cards that you deal out flip from face down (as they are in the deck) to face up (as they are once they are in play). You can use/modify the `.cardify` `ViewModifier` from lecture if you want.
4. Add any other animation you can think of that would make sense.