< Previous: Key points

Table of Contents >

Challenge

You have a basic understanding of arrays, state, views, images, text, and more, so let's put them together: your challenge is to make a brain training game that challenges players to win or lose at rock, paper, scissors.

So, very roughly:

- Each turn of the game the app will randomly pick either rock, paper, or scissors.
- Each turn the app will alternate between prompting the player to win or lose.
- The player must then tap the correct move to win or lose the game.
- If they are correct they score a point; otherwise they lose a point.
- The game ends after 10 questions, at which point their score is shown.

So, if the app chose "Rock" and "Win" the player would need to choose "Paper", but if the app chose "Rock" and "Lose" the player would need to choose "Scissors".

Hacking with Swift+ subscribers can get a complete video solution for this checkpoint here: Solution to Rock, Paper, Scissors. If you don't already subscribe, you can start a free trial today.

To solve this challenge you'll need to draw on skills you learned in tutorials 1 and 2:

1. Start with an App template, then create a property to store the three possible moves: rock, paper, and scissors.
2. You'll need to create two `@State` properties to store the app's current choice and whether the player should win or lose.
3. You can use `Int.random(in:)` to select a random move. You can use it for whether the player should win too if you want, but there's an even easier choice: `Bool.random()` is randomly true or false. After the initial value, use `toggle()` between rounds so it's always changing.
4. Create a `VStack` showing the player's score, the app's move, and whether the player should win or lose. You can use `if shouldWin` to return one of two different text views.
5. The important part is making three buttons that respond to the player's move: Rock, Paper, or Scissors.
6. Use the `font()` modifier to adjust the size of your text. If you're using emoji for the three moves, they also scale. **Tip:** You can ask for very large system fonts using `.font(.system(size: 200))` – they'll be a fixed size, but at least you can make sure they are nice and big!

I'm going to provide some hints below, but I suggest you try to complete as much of the challenge as you can before reading them.

Hints:

- Start off with the simplest logic you can: three buttons, each with logic that says "the player tapped rock, the player was trying to win, and the app chose scissors, so add 1 point."
- Once you have that working, look for a way to simplify your logic such as an array of which items beat each move. For example, if your moves array was `["Rock", "Paper", "Scissors"]` your array of winning moves would be `["Paper", "Scissors", "Rock"]`.
- You don't need to add graphics if you don't want to; just some text views and buttons is enough. Why not try emoji?

This should be a fun exercise: there's a little bit of Swift, a little bit of SwiftUI, and a little bit of logic, all wrapped up in a game where you can really go to town on theming if you want to.