Rock Paper Scissors Lizard Spock 👊🖐✌🦎🖖

Activate:

1. Have two students play RPS using their hands. Have a third student record a video of it using an iPhone or iPad.

2. Keep track of the number of ties in the game between the two players.

3. Determine the percentage of ties that occurred.

4. Discuss how can we make the game better and result in fewer ties.

5. Highlight that we want to give players more options without too many to remember.

6. Discuss five choices - students can draw a pentagon and place emoji at points to map choices.

7. Introduce RPSLS with Big Bang Theory segment from YouTube.

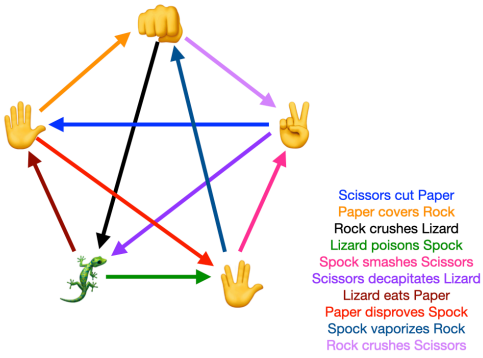
Explore:

1. Give students the game rules as a text file.

2. Have students create their own game using five different emoji and new rules.

3. Build a game graphic in Keynote using the emoji used in the game, include the game rules as text.

4. Export the graphic as an image to the .png format .



see larger graphic below

5. Students now need to decide if they want to build two-screen app (where the game rules are displayed on a second screen as image file) or a single view of the game (where the rules are displayed as a label when the UI is updated to show if the player has won, loss, or played to a draw).

Apply:

1. Create branch from original RPS app to create Rock Paper Scissors Lizard Spock app.
    1. Take original app and put buttons and labels in stack views and add constraints as you like for the design of your app (You can see settings below).
    2. If the app is put on a physical device the emoji will show a blue line under the emoji. This is because of the Text Color attribute setting in the Attributes Inspector.
    3. Update “Text Color” in the Attributes Inspector to be “Secondary System Fill Color” (You can see settings below).
    4. You can see a screencast of the this process here: Add constraints

2. Update Sign.swift enum cases and emoji variable to add lizard and Spock

3. Emphasize that we do not need to Update GameState.swift enum because there are still four states of the game: start, win, lose, draw

4. Update gameState func to include the extra wining comparisons for Rock, Paper, and Scissors.

5. Add winning cases for lizard & Spock

6. Update randomSign in Sign.swift to include lizard and Spock
    - *You can see a screencast of steps 2 - 6 here*: Updating the Models.

7. Add lizard and Spock buttons in MainStoryboard
    1. Embed in stack view
    2. Drag into larger vertical stack view
    3. Adjust main vertical stack view spacing

    (You can see settings and pictures below)

8. Create outlets for for lizard and Spock buttons

9. Create actions for lizard and Spock buttons
    - *You can see a screencast of steps 7 - 9 here*: Updating the View.

10. Update the UpdateUI function in ViewController.swift for lizard and Spock

11. Update the play function in ViewController.swift for lizard and Spock

12. Add cases for lizard and spock in userSign switch statement inside of play function

13. Add calls to the play function in lizard and Spock IBAction functions

14. Game is now complete. Build and run. Will get incorrect emoji chosen for lizard and Spock. Need to fix by deleting rockChosen connection in Connections Inspector.

15. Build and run. If everything works Checkout of brach back into main and then merge branches
    - *You can see a screencast of steps 10 - 15 here*: Updating the Controller.

16. How can we make it better
    1. Make the opponentSignLabel randomly choose an emoji from multiple emoji?
    2. Allow the user to see the game rules?

17. Random opponentSignLabel emoji
    1. Update Gamestate.swift status variable start case to include lizard and Spock.
    2. Create a variable array of emoji in viewController.swift.
    3. Assign the opponentSignLabel.text property to be a random element to the variable array in viewController.swift.
    - *You can see a screencast of step 17 here:* Updating Opponent Emoji

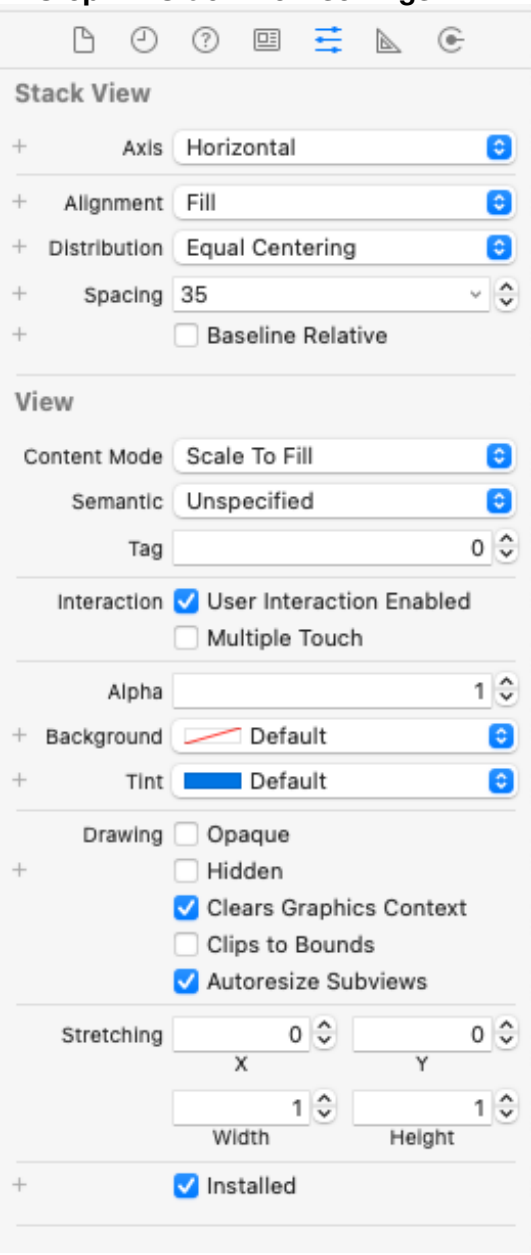
18. Create new branch for adding rules

19. Rules on a second screen
    1. Export rules Keynote as an image and save as a png file.
    2. Add a new View Controller.
    3. Add a button and constraints at bottom to view rules.
    4. Control-drag and create a show segue to new view controller.
    5. Add imageView from Object Library to new view controller.
    6. Drag Keynote image file in assets.xcassets folder.
    7. Choose your Keynote rules image from the image dropdown in the Attributes Inspector for the imageView.
    8. Embed the initial viewController in a NavigationController.
    9. Build and run. If everything works Checkout of brach back into main and then merge branches.
    - *You can see a screencast of steps 18 - 19 here:* Adding Game Rules via Image.

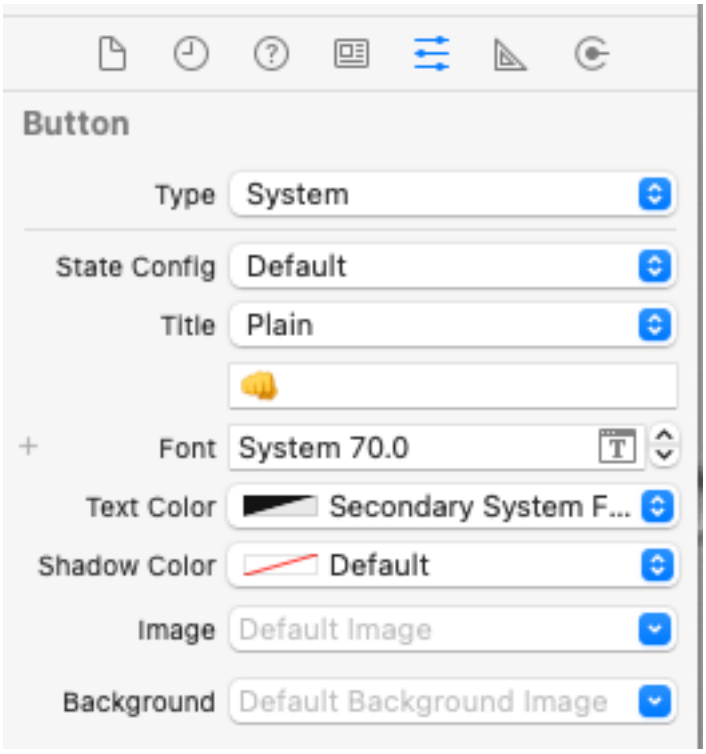
20. Rules as a label
    1. **Add new branch**
    2. Add a label to large stack view named ruleLabel and set Font to Title 2.

3. Create a rule variable in Sign.swift.
4. Assign rule to be an empty string for .draw and .lose cases in Sign.swift.
5. Assign rule as a string to each of the existing winning results with the corresponding rule in Sign.swift.
6. Assign the appropriate rule as a string and return .lose for each of the losing outcome in Sign.swift.
7. Open the Assistant Editor and create the ruleLabel outlet in ViewController.swift.
8. Assign ruleLabeL in updateUI function in .start in ViewController.swift to hide the label.
9. Assign ruleLabel to be visible in play function.
10. Assign the appropriate rule to the ruleLabel's text property.
11. Build and run. If everything works Checkout of brach back into main and then merge branches.
    - *You can see a screencast of step 20 here:* Adding Game Rules via Label

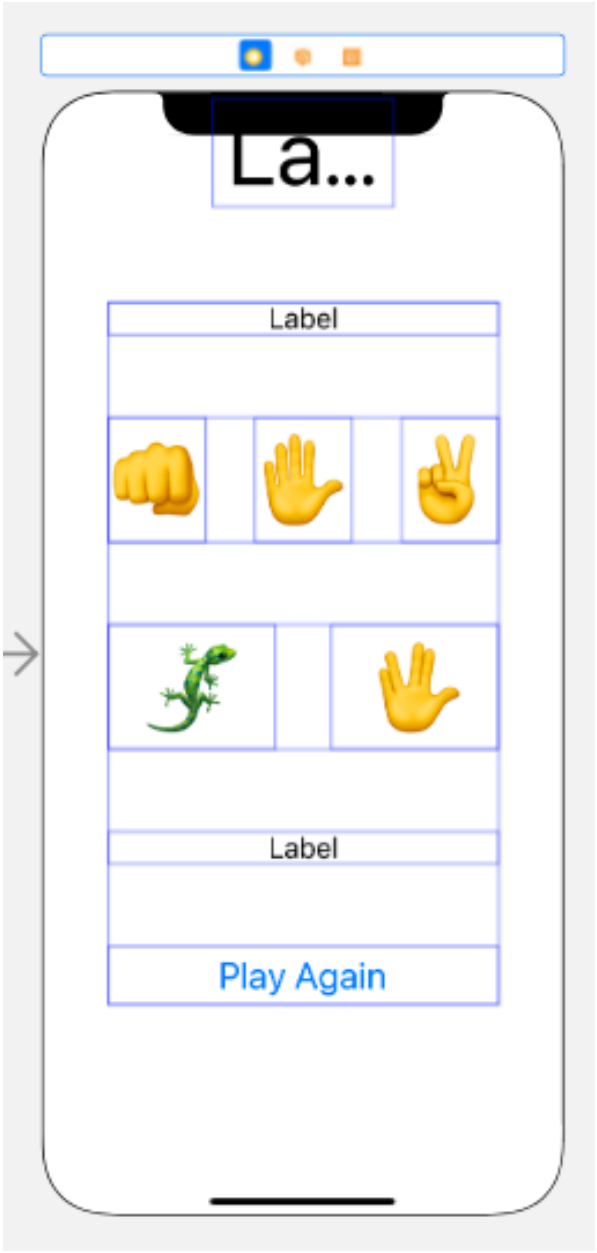
Step 1.1 Stack View settings

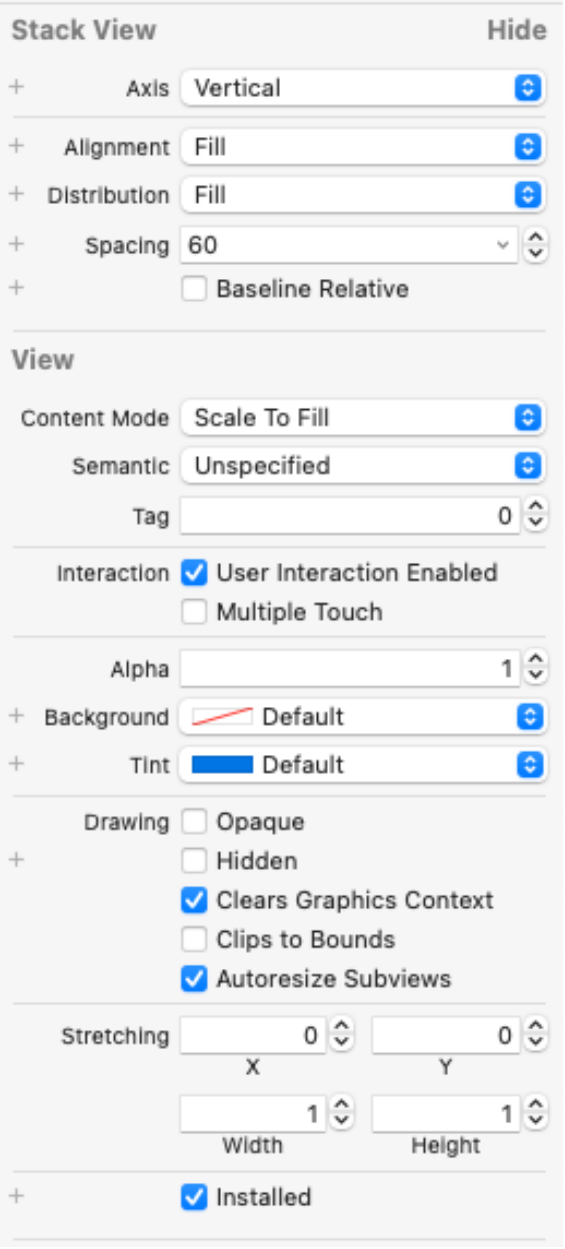


Step 1.3 Font settings

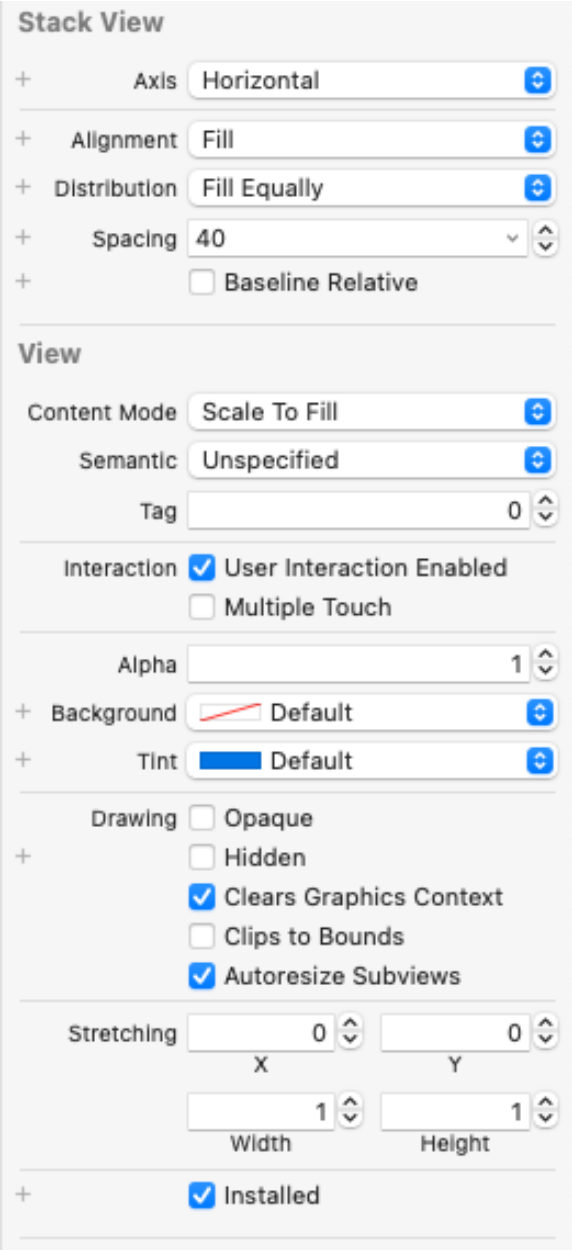


7.1 Layout and Settings

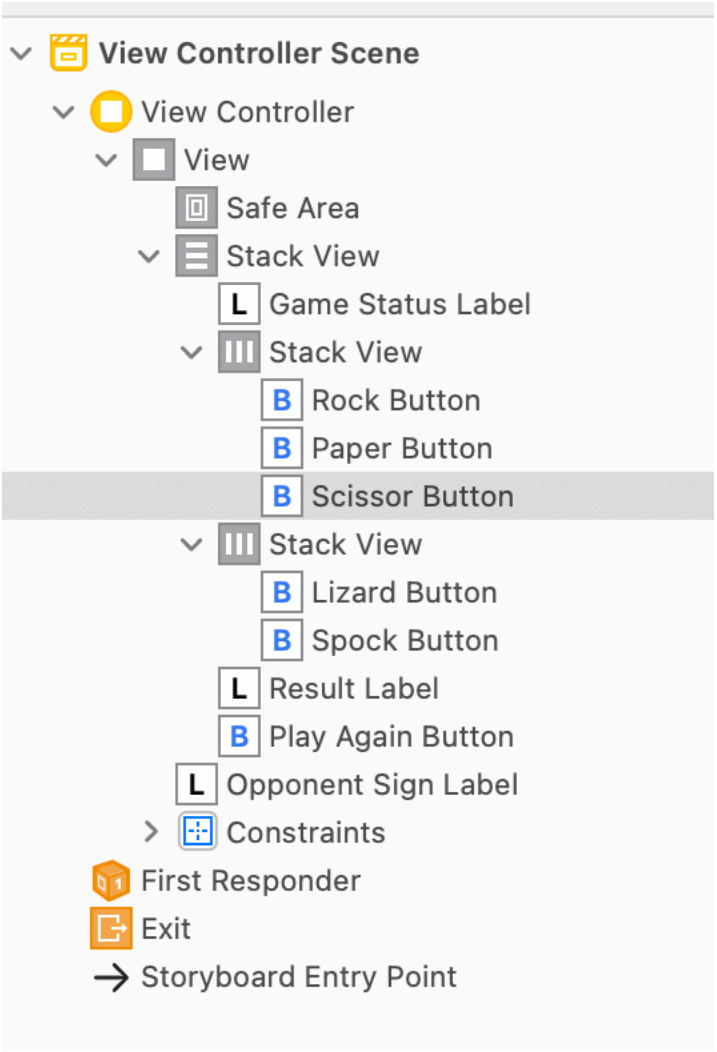


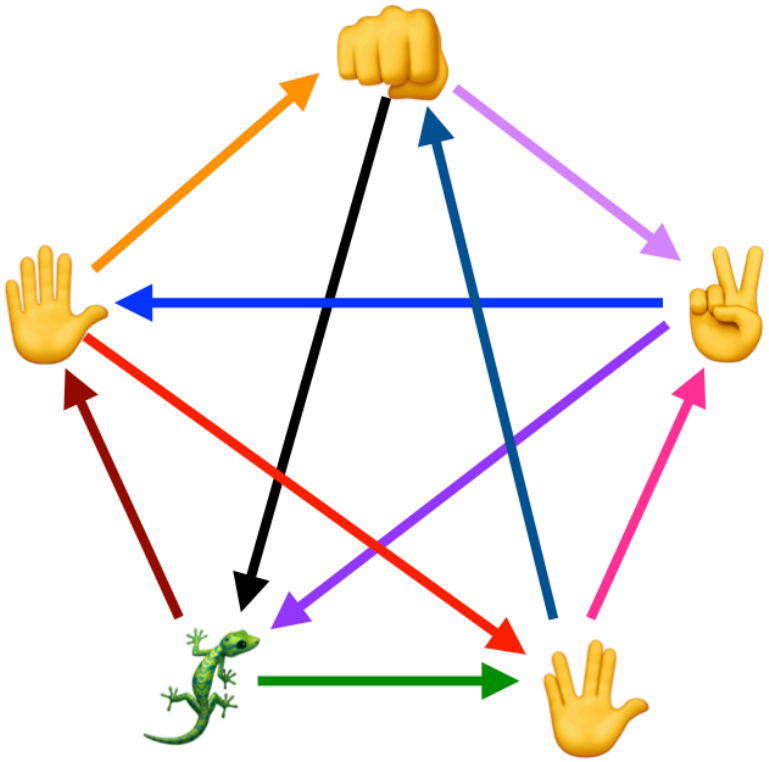


7.2 and 7.3 Settings



Document Outline for entire app





Scissors cut Paper
Paper covers Rock
Rock crushes Lizard
Lizard poisons Spock
Spock smashes Scissors
Scissors decapitates Lizard
Lizard eats Paper
Paper disproves Spock
Spock vaporizes Rock
Rock crushes Scissors