**Mobile Computing – iOS Fall’23**

**Assignment02**

**30 Points**

**Please follow the following instructions to complete this assignment.**

1. Open Xcode from the launchpad of your Mac. Clone your private GitHub repository.
2. In that repo, create a new Xcode project. Select the iOS platform and click on the App template.
3. Click on next, which will prompt you to choose options for the project.
4. Provide product name as **SimonGameApp**, “**edu.northwest.fall23.cs44653**” for organization identifier, “**Storyboard**” as interface and **Swift** as language.
5. Click on next and select an appropriate location to save your app and click on Create. A project directory will be loaded.
6. From the project navigator click on “Main.storyboard” file, a blank mobile screen will be loaded, where the required components for the app are added.

**The View**

Table 1: UI elements configuration

|  |  |  |
| --- | --- | --- |
| **UI element** | **Purpose** | **Outlet/action name** |
| 6 UILabel elements | To display “Simon Game™️” | n/a |
| To display “Score” | n/a |
| To display “Level” | n/a |
| To display the score of the player | scoreLBL |
| To display the level number | levelLBL |
| To display the message | messageLBL |
| 6 UIButtons | 4 colored buttons (Red, Blue, Green, Yellow) | coloredBtnCLCTN |
| onClickColorBtn: |
| Button Start to start the game | startBTN |
| onStart: |
| Button Reset to reset the game | onReset: |

*Note: Names that are ending with a colon (****:****) are actions.*

1. Open library (cmd+shift+l). Search for “label” and add (drag and drop) it to the storyboard.

Graphical user interface, text, application

Description automatically generated

1. Similarly, add 5 more labels to the storyboard or copy a few more labels.
2. For the header label, set the background color to the matching color in document outline (i.e., System Purple Color) and change the label text to “Simon Game™️”.
3. Moreover, set its properties as mentioned here.
   1. Color: White Color
   2. Font: Large Title
   3. Alignment: Center
4. Use 2 labels to display “Score”, and “Level”, respectively. Leave the 3 other labels to display the score, level, and message, respectively.
5. In the library, search for “button” and add 4 normal buttons and 2 tinted buttons to the storyboard.
6. Change the fill color of the normal buttons to “System Red Color”, “System Blue Color”, “System Green Color” and “System Yellow Color” respectively.
   1. Give each button a unique tag value.
   2. Initially, set alpha value for all buttons to 0.5
   3. Add all the button to an outlet collection.
7. Name the 2 tinted buttons as “Start” and “Reset”, respectively.

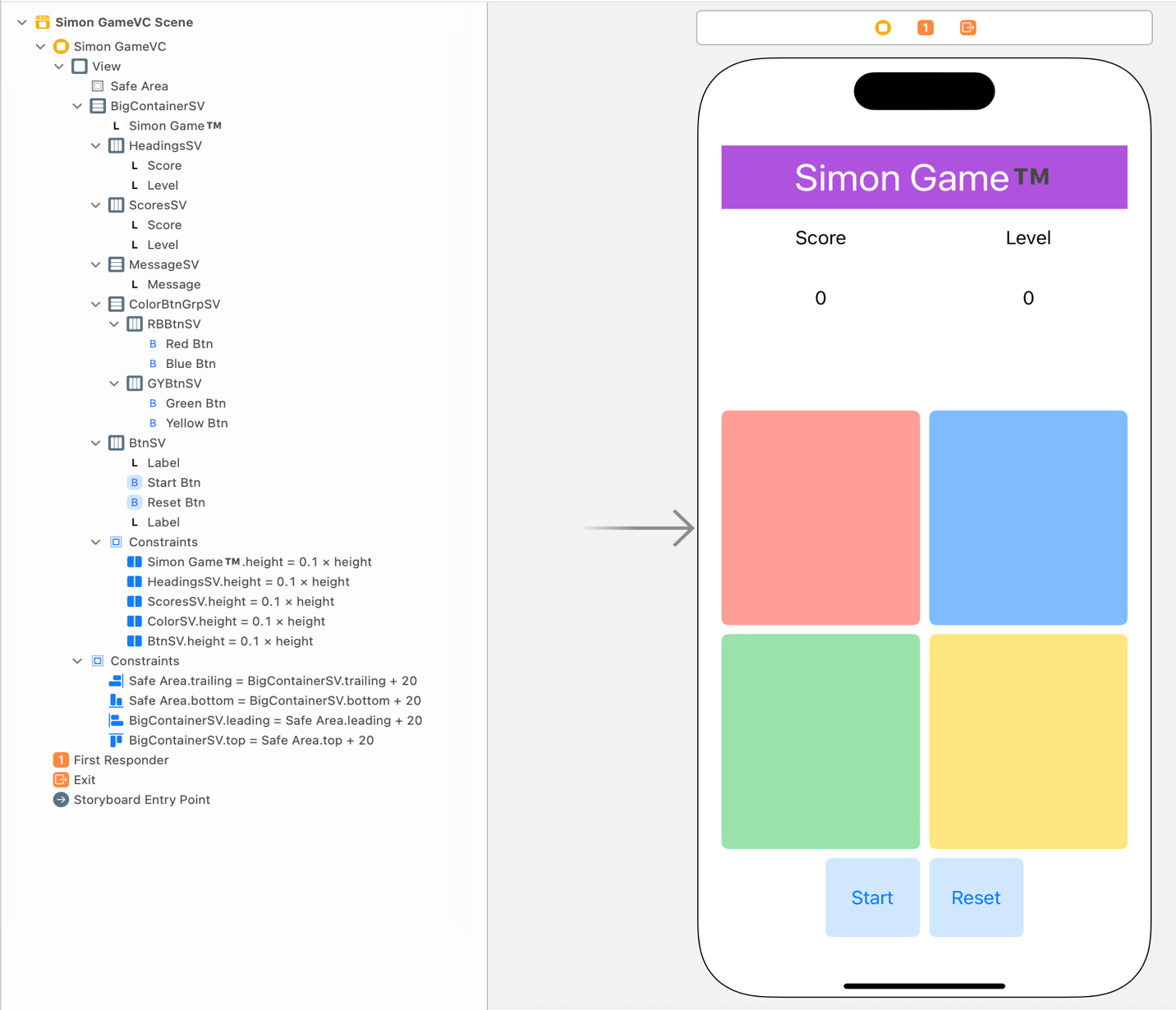


Figure Auto Layout

1. Now, all the required UI elements for the app are added to the storyboard. Apply auto layout to the app by adding constraints to UI elements as shown in Figure 1.
   1. Hint: Follow the view hierarchy in the Figure 1 (i.e., document outline on the left side) to add and organize stack views and embed elements inside them.

Table 1 SimonGameVC’s properties for stack views

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stack View** | **Axis** | **Alignment** | **Distribution** | **Spacing** |
| BigContainerSV, MessageSV | Vertical | Fill | Fill | Standard |
| HeadingsSV, ScoresSV,  RBBtnSV, GYBtnSV, BtnSV | Horizontal | Fill | Fill Equally | Standard |

**The Controller:**

1. Create a Cocoa Touch Class “SimonGameVC” that is a sub class of UIViewController and assign it as class to Main.storyboard file.
2. Create outlets and actions as specified in Table 1.
3. The Simon Game is a memory game, the player must remember the order of colors generated by the simulator and must duplicate them via taps. If the order has gone wrong, the player lost the game.
4. The game highlights a random color (red, green, blue, yellow) for each level. The player must memorize the order of the highlighted colors from the previous levels and add the color of current level to the order.
5. The game starts when the player taps the “Start” button.
6. First for the Level 1, the simulator highlights one of the colored buttons. Then it is the player’s chance to tap the highlighted button in the UI.
7. Next for the Level 2, the simulator highlights one of the colored buttons as done in the previous level. Then it is the player’s chance to duplicate the order of highlighted buttons starting from Level 1. The game goes on to higher levels.
8. For example:

Level 1: Simulator highlighted “Red” button.

Score 0: Player must duplicate the order by tapping “Red” button.

Otherwise, the game is over.

Level 2: (if player did tap the correct button in Level 1)

Score 1: Simulator highlights a colored button “Blue”.

Player must duplicate the order from Level 1. This implies the player order must be “Red” -> “Blue”.

Otherwise, the game is over.

Level 3: (if player did tap the correct buttons)

Score 2: Simulator highlights a colored button “Yellow”.

Player must duplicate the order from starting level. This implies the player order must be “Red” -> “Blue ” -> “Yellow”.

Otherwise, the game is over.

1. For every tap on the buttons, the player’s order will be verified with the original order generated by the simulator.
   1. If the match is correct, then the player’s score will be incremented by 1 and the level will be up by 1. Continue the game.
   2. If the match is incorrect, display the text “Game over!!! Lost in Level # 🎯” using the messageLBL. Replace “#” with the level.
2. If the player taps the “Reset” button, the game will get back to its initial state.

**Functionality**

1. At first, set the values of score and level to zero. Set the text messageLBL to an empty string. Disable all the colored buttons.
2. When the player clicks on Start,
   1. Enable all the colored buttons and disable the start button.
   2. Increment the level by 1.
   3. Highlight a colored button among them. (Hint: Generate a random number between 0 and 4 to highlight a colored button every time). Use the below code to animate (or highlight) the colored button.



* 1. Store the original order generated by the simulator to compare with the player’s order.

1. The player must make the number of taps equal to the level number. For each tap, check the color with the original order. (Hint: Use indexing, to check the current tap of the player is correct or not.)
   1. For example, if the level is 5, then the player must make 5 correct taps to proceed to the next level.
   2. Consider the original order as “Red” -> “Blue” -> “Blue” -> “Green” -> “Yellow”.
   3. When the player starts tapping the buttons,
      1. Red must be the first click. If correct, let the player proceed.
      2. Blue must be the second click. If correct, let the player proceed.
      3. Blue must be the third click. If the player clicks “Green”, end the game by displaying the message “Game over!!! Lost in Level 5 🎯” in the messageLBL.
   4. If the player’s order is correct, increment the level, score, and repeat the process.
2. On tapping the “Reset” button, the game gets back to the initial state.

**Submission:** Push your entire Xcode project to your private GitHub repo and submit your private repository link on the Canvas via Text Entry.