

```

1 //
2 // ColourSelectorApp.swift
3 // ColourSelector
4 //
5
6 import SwiftUI
7
8 @main
9 struct ColourSelectorApp: App {
10
11     var body: some Scene {
12         WindowGroup {
13             BrowsingView()
14         }
15     }
16 }
17
18
19 //
20 // Functions.swift
21 // ColourSelector
22 //
23
24 import Foundation
25
26 func filtering(originalList: [Palette], on desiredFloor: Hue) -> [Palette] {
27
28     if desiredFloor == .allHues {
29         return originalList
30     }
31
32     let desiredCeiling = desiredFloor.rawValue + 60.0
33     let desiredHueRange = desiredFloor.rawValue...desiredCeiling
34
35     var filteredListOfPalettes: [Palette] = []
36
37     for palette in originalList {
38
39         if desiredHueRange.contains(palette.base.hue) {
40             filteredListOfPalettes.append(palette)
41         }
42     }
43
44     return filteredListOfPalettes
45 }
46
47 }
48
49
50 //
51 // Hue.swift
52 // ColourSelector
53 //
54
55 import Foundation
56
57 enum Hue: Double {
58     case allHues = -30
59     case red = 0
60     case yellow = 60
61     case green = 120
62     case teal = 180
63     case blue = 240

```

```

64     case purple = 300
65 }
66
67
68 //
69 // Tile.swift
70 // ColourSelector
71 //
72
73 import SwiftUI
74
75 struct Tile: Identifiable {
76
77     let id = UUID()
78     let hue: Double
79     let saturation: Double = 80
80     let brightness: Double
81
82     var hueFormattedAsString: String {
83         return "\(hue.formatted(.number.precision(.fractionLength(1))))°"
84     }
85
86     var color: Color {
87
88         return Color(
89             hue: hue / 360.0,
90             saturation: saturation / 100.0,
91             brightness: brightness / 100.0
92         )
93     }
94 }
95
96 }
97
98 let redTile = Tile(hue: 0.0, brightness: 90)
99 let yellowTile = Tile(hue: 60.0, brightness: 90)
100 let blueTile = Tile(hue: 240.0, brightness: 90)
101
102
103 //
104 // Palette.swift
105 // ColourSelector
106 //
107
108 import SwiftUI
109
110 struct Palette: Identifiable {
111
112     let id = UUID()
113     let base: Tile
114
115     var darker: Tile {
116         return Tile(hue: base.hue, brightness: base.brightness - 30.0)
117     }
118     var darkest: Tile {
119         return Tile(hue: base.hue, brightness: base.brightness - 60.0)
120     }
121 }
122 }
123
124 let examplePalette = Palette(
125     base: Tile(hue: 0.0, brightness: 90.0)
126 )

```

```
127
128
129 //
130 // TileView.swift
131 // ColourSelector
132 //
133
134 import SwiftUI
135
136 struct TileView: View {
137
138     var tile: Tile
139     var size: Double
140
141     var body: some View {
142
143         Rectangle()
144             .fill(tile.color)
145             .frame(width: size, height: size)
146
147     }
148 }
149
150 #Preview {
151     TileView(tile: blueTile, size: 100)
152 }
153
154
155 //
156 // PaletteView.swift
157 // ColourSelector
158 //
159
160 import SwiftUI
161
162 struct PaletteView: View {
163
164     let palette: Palette
165
166     var body: some View {
167
168         HStack(spacing: 0) {
169
170             TileView(
171                 tile: palette.base,
172                 size: 50
173             )
174
175             TileView(
176                 tile: palette.darker,
177                 size: 50
178             )
179
180             TileView(
181                 tile: palette.darkest,
182                 size: 50
183             )
184
185             Spacer()
186
187         }
188
189     }
```

```

190 }
191
192 #Preview {
193     PaletteView(palette: examplePalette)
194 }
195
196 //
197 // BrowsingView.swift
198 // ColourSelector
199 //
200 //
201
202 import SwiftUI
203
204 struct BrowsingView: View {
205
206     @State var selectedBaseHue = 0.0
207
208     @State var history: [Palette] = []
209
210     @State var selectedHueRange: Hue = .allHues
211
212     var tileCreatedFromSelectedHue: Tile {
213         return Tile(hue: selectedBaseHue, brightness: 90)
214     }
215
216     var paletteCreatedFromBaseTile: Palette {
217         return Palette(base: tileCreatedFromSelectedHue)
218     }
219
220     var body: some View {
221
222         HStack {
223
224             VStack(alignment: .leading, spacing: 20) {
225
226
227                 TileView(
228                     tile: tileCreatedFromSelectedHue,
229                     size: 150.0
230                 )
231                 .padding(.trailing)
232
233                 Text(tileCreatedFromSelectedHue.hueFormattedAsString)
234                     .font(.largeTitle)
235
236                 Slider(value: $selectedBaseHue,
237                     in: 0...360,
238                     label: { Text("Base Hue") },
239                     minimumValueLabel: { Text("0") },
240                     maximumValueLabel: { Text("360") })
241
242                 HStack {
243
244                     PaletteView(palette: paletteCreatedFromBaseTile)
245
246                     Spacer()
247
248                     Button(action: {
249                         savePalette()
250                     }, label: {
251                         Text("Save")
252

```

```

253         .font(.subheadline.smallCaps())
254     })
255     .buttonStyle(.bordered)
256
257 }
258
259     Spacer()
260
261 }
262
263     VStack(alignment: .leading) {
264
265         Picker("Filtering on", selection: $selectedHueRange) {
266             Text("All hues (no filtering)").tag(Hue.allHues)
267             Text("Red to yellow (0° to 60°)").tag(Hue.red)
268             Text("Yellow to green (60° to 120°)").tag(Hue.yellow)
269             Text("Green to teal (120° to 180°)").tag(Hue.green)
270             Text("Teal to blue (180° to 240°)").tag(Hue.teal)
271             Text("Blue to purple (240° to 300°)").tag(Hue.blue)
272             Text("Purple to red (300° to 360°)").tag(Hue.purple)
273         }
274         .padding(10)
275
276         Image("IllustrationOfHSBColorModel")
277             .resizable()
278             .scaledToFit()
279             .frame(width: 200)
280
281         List(
282             filtering(
283                 originalList: history,
284                 on: selectedHueRange
285             )
286         ) { currentPalette in
287             PaletteView(palette: currentPalette)
288         }
289
290     }
291
292 }
293     .padding()
294 }
295
296 func savePalette() {
297     history.append(paletteCreatedFromBaseTile)
298 }
299 }
300
301 #Preview {
302
303     BrowsingView()
304         .frame(height: 600)
305
306 }
307

```