```
2c2
< // Swift source code of main.swift Version 8.1.3 at greenchess.com //
---
> // Swift source code of main.swift Version 8.1.0 at greenchess.com //
1300,1304c1300,1301
<     // the window scale //
<     var Scale:CGFloat = 2
<
<     // the hue layer //
<     var HueLayer:CGLayer!
---
>     // whether view has a border or not //
>     var Bordered:Bool = false
1324,1325c1321,1322
<     // the current hue rectangle //
<     var CurrentHueRect:NSRect = NSRect()
---
>     // the current background color set //
>     var ColorSet:Int = 0
1327,1328c1324,1325
<     // the color hues //
<     var Hues:[NSColor] = []
---
>     // the current color rectangle //
>     var CurrentColorRect:NSRect = NSRect()
1330,1331c1327,1328
<     // the current hue //
<     var CurrentHue:Int = 0
---
>     // the colors //
>     var Colors:[NSColor] = []
1333,1334c1330,1331
<     // the previous hue //
<     var PreviousHue:Int = -1
---
>     // the current color //
>     var CurrentColor:Int = 0
1337c1334
<     var Width:CGFloat = 0
---
>     var Width:Float = 0
1345,1360c1342,1343
<     // the maximum hue index //
<     func MaxHue() -> Int
<     {
<         // return the maximum hue index //
<         return 122
<     }
<
<     // the current hue //
<     func GetCurrentHue() -> NSColor
<     {
<         // return the current hue
<         return Hues[CurrentHue]
<     }
<
<     // update the current hue //
<     func UpdateCurrentHue(_ Event:NSEvent)
---
>     // update the current color //
>     func UpdateCurrentColor(_ Event:NSEvent)
1365,1379c1348,1349
<         // compute the current hue //
<         ComputeCurrentHue()
<
<         // if the previous hue equals the current hue //
<         if PreviousHue == CurrentHue
<         {
<             // return since the chosen hue hasn't changed //
<             return
<         }
<         // have chosen a new hue //
<         else
<         {
<             // set the previous hue to the current hue //
<             PreviousHue = CurrentHue
<         }
---
>         // get the current color //
>         GetCurrentColor()
1423,1424c1393,1394
<             // set the window background color //
<             Board.SetBackgroundColor()
---
>             // set the current background color //
>             Board.SetBackgroundColor(ColorSet)
1429,1431d1398
<
<         // update the view //
<         displayIgnoringOpacity(bounds)
1437,1438c1404,1405
<         // update the current hue from the mouse location //
<         UpdateCurrentHue(Event)
---
>         // update the current color from the mouse location //
>         UpdateCurrentColor(Event)
1444,1452c1411,1412
<         // update the current hue from the mouse location //
<         UpdateCurrentHue(Event)
<     }
<
<     // override mouse up //
<     override func mouseUp(with Event:NSEvent)
<     {
<         // update the current hue from the mouse location //
<         UpdateCurrentHue(Event)
---
>         // update the current color from the mouse location //
>         UpdateCurrentColor(Event)
1455,1456c1415,1416
<     // compute the current hue //
<     func ComputeCurrentHue()
---
>     // get the current color //
>     func GetCurrentColor()
1458,1459c1418,1423
<         // get the current hue from the mouse location //
<         CurrentHue = Int((Mouse.x - 30) / Width)
---
>         // if not a color set that was not previously active //
>         if !(ColorSet > 0 && !Bordered)
>         {
>             // get the current color from the mouse location //
>             CurrentColor = Int(Float(Mouse.x - 30) / Width)
>         }
1461,1462c1425,1426
<         // if the current hue is less than zero //
<         if CurrentHue < 0
---
>         // if the current color is less than zero //
>         if CurrentColor < 0
1464,1465c1428,1429
<             // set the current hue to zero //
<             CurrentHue = 0
---
>             // set the current color to zero //
>             CurrentColor = 0
1467,1468c1431,1432
<         // if the current hue is greater than the maximum possible value //
<         else if CurrentHue > Hues.count - 1
---
>         // if the current color is greater than the maximum possible value //
>         else if CurrentColor > Colors.count - 1
1470,1471c1434,1435
<             // set the current hue to the maximum possible value //
<             CurrentHue = Hues.count - 1
---
>             // set the current color to the maximum possible value //
>             CurrentColor = Colors.count - 1
1478,1479c1442,1457
<         // draw the hue layer into the current context //
<         NSGraphicsContext.current?.cgContext.draw(HueLayer, in:bounds)
---
>         // loop over all the colors //
>         for Index in 0 ..< Colors.count
>         {
>             // set the current fill color //
>             Colors[Index].setFill()
>
>             // draw the current color rectangle //
>             NSBezierPath(rect:NSMakeRect(CGFloat(Float(Index) * Width), 10, CGFloat(Width), 20))
>                 .fill()
>         }
>
>         // set the border color //
>         NSColor(red:0, green:0, blue:0, alpha:0.4).set()
>
>         // draw the colors spectrum border //
>         NSBezierPath(rect:NSMakeRect(1, 10, frame.size.width - 2, 20)).stroke()
1481,1482c1459,1460
<         // if the current hue is zero //
<         if CurrentHue == 0
---
>         // if the current color is zero //
>         if CurrentColor == 0
1484,1485c1462,1463
<             // set the first current hue rectangle //
<             CurrentHueRect = NSMakeRect(0.5, 1, Width, 38)
---
>             // set the first current color rectangle //
>             CurrentColorRect = NSMakeRect(0.5, 1, CGFloat(Width), 38)
1487,1488c1465,1466
<         // if the current hue is the last hue //
<         else if CurrentHue == Hues.count - 1
---
>         // if the current color is the last color //
>         else if CurrentColor == Colors.count - 1
1490,1491c1468,1470
<             // set the last current hue rectangle //
<             CurrentHueRect = NSMakeRect(CGFloat(CurrentHue) * Width - 0.5, 1, Width, 38)
---
>             // set the last current color rectangle //
>             CurrentColorRect = NSMakeRect(CGFloat(Float(CurrentColor) * Width - 1), 1, CGFloat(Width),
>                                           38)
1493c1472
<     // is between the first and last possible hue //
---
>     // is between the first and last possible color //
1496,1497c1475,1476
<             // set the current hue rectangle //
<             CurrentHueRect = NSMakeRect(CGFloat(CurrentHue) * Width, 1, Width, 38)
---
>             // set the current color rectangle //
>             CurrentColorRect = NSMakeRect(CGFloat(Float(CurrentColor) * Width), 1, CGFloat(Width), 38)
1500,1501c1479,1480
<         // fill the hue to the current hue //
<         Hues[CurrentHue].setFill()
---
>         // set the fill color to the current color //
>         Colors[CurrentColor].setFill()
1503,1504c1482,1483
<         // fill the current hue rectangle with the current hue //
<         NSBezierPath(rect:CurrentHueRect).fill()
---
>         // fill the current color rectangle with the current color //
>         NSBezierPath(rect:CurrentColorRect).fill()
1507,1548c1486
<         NSColor.black.set()
<
<         // draw the current hue rectangle border //
<         NSBezierPath(rect:CurrentHueRect).stroke()
<     }
<
<     // initialize the color object //
<     func Initialize(_ ChosenHue:Int)
<     {
<         // look over all the hues //
<         for Index in 0 ..< 120
<         {
<             // if index is 0 //
<             if Index == 0
<             {
<                 // append black to the hues //
<                 Hues.append(NSColor.black)
<             }
<             // if index is 40 //
<             else if Index == 40
<             {
<                 // append light gray to the hues //
<                 Hues.append(NSColor.lightGray)
<             }
<             // if index is 80 //
<             else if Index == 80
<             {
<                 // append white to the hues //
<                 Hues.append(NSColor.white)
<             }
<
<             // create a specific hue //
<             Hues.append(NSColor(hue:CGFloat(Index % 40) / 39.0, saturation:0.75, brightness:
<                         CGFloat(Index) / 160 + 0.25, alpha:1))
<         }
<
<         // create the hue layer //
<         HueLayer = CGLayer(CGContext(data:nil, width:Int(Scale * frame.size.width), height:
<                            Int(Scale * frame.size.height), bitsPerComponent:8, bytesPerRow:0,
<                            space:CGColorSpaceCreateDeviceRGB(), bitmapInfo:CGImageAlphaInfo
<                            .noneSkipFirst.rawValue)!, size:CGSize(width:Scale * frame.size.width,
<                            height:Scale * frame.size.height), auxiliaryInfo:nil)!
---
>         NSColor(red:0, green:0, blue:0, alpha:0.4).set()
1550,1551c1488,1489
<         // set the current hue //
<         CurrentHue = ChosenHue
---
>         // draw the current color rectangle border //
>         NSBezierPath(rect:CurrentColorRect).stroke()
1553,1563c1491,1492
<         // set the current hue bar width //
<         Width = frame.size.width / CGFloat(Hues.count)
<
<         // set the current graphics context to be the hue layer context //
<         NSGraphicsContext.current = NSGraphicsContext(cgContext:HueLayer.context!, flipped:false)
<
<         // set the current graphics context alpha to one //
<         NSGraphicsContext.current?.cgContext.setAlpha(1)
<
<         // loop over all the hues //
<         for Index in 0 ..< Hues.count
---
>         // if have a border //
>         if Bordered
1565,1570c1494,1495
<             // set the current fill hue //
<             Hues[Index].setFill()
<
<             // draw the current hue rectangle //
<             NSBezierPath(rect:NSMakeRect(Scale * Width * CGFloat(Index), Scale * 10, Scale * Width,
<                          Scale * 20)).fill()
---
>             // draw the view border //
>             NSBezierPath(rect:bounds).stroke()
1571a1497,1503
>     }
>
>     // store the chosen color set and color //
>     func StoreColorSet(_ TheColors:[NSColor], ChosenColor:Int)
>     {
>         // store the chosen color set //
>         Colors = TheColors
1573,1574c1505,1506
<         // set the border color //
<         NSColor.black.set()
---
>         // set the current color //
>         CurrentColor = ChosenColor
1576,1578c1508,1509
<         // draw the hue layer border //
<         NSBezierPath(rect:NSMakeRect(0.5 * Scale, Scale * 10, Scale * (frame.size.width - 1), Scale
<                      * 20)).stroke()
---
>         // set the current color bar width //
>         Width = Float(frame.size.width) / Float(Colors.count)
1976,1978d1906
<     // the snapshot image representation //
<     var SnapshotImageRep:NSBitmapImageRep!
<
1990a1919,1921
>
>     // the board roughness cg layer //
>     var CGLayerBoardRoughness:CGLayer!
1993a1925,1927
>
>     // the tint overlay cg layer //
>     var CGLayerTintOverlay:CGLayer!
1996a1931,1933
>
>     // the tint overlay opacity //
>     var TintOverlayOpacity:CGFloat = 0
2048,2050d1984
<     // the diagram layer for drawing //
<     var DiagramLayer:CGLayer!
<
2156,2161d2089
<     // board roughness texture image //
<     var BoardRoughnessTexture:NSImage = NSImage()
<
<     // board roughness texture color //
<     var BoardRoughnessTextureColor:NSColor = NSColor()
<
2168,2169c2096,2097
<     // the square roughness circle //
<     var RoughnessCircle:NSBezierPath = NSBezierPath(ovalIn:NSMakeRect(1, 1, 1, 1))
---
>     // the square roughness circles //
>     var RoughnessCircles:NSBezierPath = NSBezierPath()
2171,2172c2099,2103
<     // set that not initially a valid diagram path //
<     var ValidPath:Bool = false
---
>     // set that not initially a valid shape //
>     var ValidShape:Bool = false
>
>     // the shape layers //
>     var Shapes:[CAShapeLayer] = []
2174,2175c2105,2106
<     // a list of diagram paths for drawing //
<     var DiagramPaths:[CGMutablePath] = []
---
>     // a shape layer //
>     var Shape:CAShapeLayer = CAShapeLayer()
2177,2178c2108,2109
<     // a list of diagram path colors //
<     var DiagramPathColors:[CGColor] = []
---
>     // a shape path //
>     var ShapePath:CGMutablePath = CGMutablePath()
2180,2181c2111,2112
<     // the window texture x index //
<     var TextureX:Int = 0
---
>     // the board roughness x index //
>     var RoughnessXIndex:Int = 0
2183,2184c2114,2115
<     // the window texture y index //
<     var TextureY:Int = 0
---
>     // the board roughness y index //
>     var RoughnessYIndex:Int = 0
2188a2123,2131
>
>     // the light square roughness image //
>     var SquareRoughnessLight:NSImage = NSImage()
>
>     // the board tint white image //
>     var BoardTintWhiteImage:NSImage = NSImage()
>
>     // the board tint black image //
>     var BoardTintBlackImage:NSImage = NSImage()
>
>     // the dark square roughness image //
>     var SquareRoughnessDark:NSImage = NSImage()
2263a2207,2209
>
>     // the window scaling value //
>     var ColorScaling:CGFloat = 0
2348,2349c2294,2295
<     // the current label hue index //
<     var LabelHue:Int = 0
---
>     // the current label color index //
>     var LabelColor:Int = 0
2351,2352c2297,2298
<     // the current info hue index //
<     var InfoHue:Int = 0
---
>     // the current info color index //
>     var InfoColor:Int = 0
2354,2355c2300,2301
<     // the current move list hue index //
<     var MoveListHue:Int = 0
---
>     // the current move list color index //
>     var MoveListColor:Int = 0
2357,2358c2303,2304
<     // the current game list hue index //
<     var GameListHue:Int = 0
---
>     // the current game list color index //
>     var GameListColor:Int = 0
2360,2361c2306,2307
<     // the game clock border hue index //
<     var GameClockHue:Int = 0
---
>     // the game clock border color index //
>     var GameClockColor:Int = 0
2363,2364c2309,2310
<     // the board tint hue //
<     var BoardTintHue:Int = 0
---
>     // the board tint color //
>     var BoardTintColor:Int = 0
2366,2367c2312,2313
<     // the game clock background hue index //
<     var GameClockBackgroundHue:Int = 0
---
>     // the game clock background color index //
>     var GameClockBackgroundColor:Int = 0
2446,2447c2392,2393
<     // the chess board border texture color //
<     var BorderTextureColor:Int = 0
---
>     // the chess board border texture //
>     var BorderTexture:Int = 0
2497a2444,2473
>     // the rainbow background colors //
>     var RainbowBackgroundColors:[NSColor] = []
>
>     // the blue background colors //
>     var BlueBackgroundColors:[NSColor] = []
>
>     // the green background colors //
>     var GreenBackgroundColors:[NSColor] = []
>
>     // the yellow background colors //
>     var YellowBackgroundColors:[NSColor] = []
>
>     // the gray background colors //
>     var GrayBackgroundColors:[NSColor] = []
>
>     // the brown background colors //
>     var BrownBackgroundColors:[NSColor] = []
>
>     // the teal background colors //
>     var TealBackgroundColors:[NSColor] = []
>
>     // the font colors //
>     var FontColors:[NSColor] = []
>
>     // the game clock colors //
>     var GameClockColors:[NSColor] = []
>
>     // the board tint colors //
>     var BoardTintColors:[NSColor] = []
2501,2503d2476
<     // the board tint hues //
<     var BoardTintHues:[NSColor] = []
<
2530,2538d2502
<
<     // the movie size //
<     var MovieSize:Int = 1
<
<     // the movie frames per second from the slider //
<     var MovieFPSFromSlider:Int = 1
<
<     // the movie frames per second //
<     var MovieFPS:Float = 1
2542,2544c2506,2529
<
<     // the current window background hue //
<     var WindowBackgroundHue:Int = 0
---
>
>     // the current background color set //
>     var BackgroundColorSet:Int = 0
>
>     // the current rainbow background color //
>     var RainbowBackgroundColor:Int = 0
>
>     // the current blue background color //
>     var BlueBackgroundColor:Int = 0
>
>     // the current brown background color //
>     var BrownBackgroundColor:Int = 0
>
>     // the current green background color //
>     var GreenBackgroundColor:Int = 0
>
>     // the current teal background color //
>     var TealBackgroundColor:Int = 0
>
>     // the current yellow background color //
>     var YellowBackgroundColor:Int = 0
>
>     // the current gray background color //
>     var GrayBackgroundColor:Int = 0
2934,2939d2918
<     // the diagram path location //
<     var PathLocation:NSPoint = CGPoint()
<
<     // the previous mouse location //
<     var PreviousMouseLocation:NSPoint = NSPoint()
<
3181a3161,3163
>
>     // the board snapshot bitmap image //
>     var SnapshotBitmap:NSBitmapImageRep!
3201,3203d3182
<     // the window color preview image //
<     var WindowColorPreviewImage:NSImage = NSImage()
<
3216,3217c3195,3214
<     // the window background colors view //
<     @IBOutlet var WindowBackgroundColorView:ColorsView!
---
>     // the rainbow background colors view //
>     @IBOutlet var RainbowBackgroundView:ColorsView!
>
>     // the blue background colors view //
>     @IBOutlet var BlueBackgroundView:ColorsView!
>
>     // the yellow background colors view //
>     @IBOutlet var YellowBackgroundView:ColorsView!
>
>     // the gray background colors view //
>     @IBOutlet var GrayBackgroundView:ColorsView!
>
>     // the teal background colors view //
>     @IBOutlet var TealBackgroundView:ColorsView!
>
>     // the green background colors view //
>     @IBOutlet var GreenBackgroundView:ColorsView!
>
>     // the brown background colors view //
>     @IBOutlet var BrownBackgroundView:ColorsView!
3546,3551d3542
<     // the movie size slider //
<     @IBOutlet var MovieSizeSlider:NSSlider!
<
<     // the movie frames per second slider //
<     @IBOutlet var MovieFPSSlider:NSSlider!
<
3672,3677d3662
<     // the text field for the movie size view //
<     @IBOutlet var MovieSizeView:NSTextField!
<
<     // the text field for the movie frames per second //
<     @IBOutlet var MovieFPSView:NSTextField!
<
3732,3733c3717,3718
<     // the window background colors text field //
<     @IBOutlet var WindowColorsBackgroundField:NSTextField!
---
>     // the rainbow colors text field //
>     @IBOutlet var RainbowBackgroundField:NSTextField!
3740a3726,3743
>     // the teal colors text field //
>     @IBOutlet var TealBackgroundField:NSTextField!
>
>     // the blue colors text field //
>     @IBOutlet var BlueBackgroundField:NSTextField!
>
>     // the green colors text field //
>     @IBOutlet var GreenBackgroundField:NSTextField!
>
>     // the yellow colors text field //
>     @IBOutlet var YellowBackgroundField:NSTextField!
>
>     // the gray colors text field //
>     @IBOutlet var GrayBackgroundField:NSTextField!
>
>     // the brown colors text field //
>     @IBOutlet var BrownBackgroundField:NSTextField!
>
3979c3982
<                                     CGFloat(WindowTextureOpacity) / 150.0)
---
>                                     CGFloat(WindowTextureOpacity) / 100.0)
3986,4000d3988
<
<       // create the window color preview image //
<       WindowColorPreviewImage = NSImage(size:NSSize(width:200, height:200))
<
<       // lock the window color preview image focus //
<       WindowColorPreviewImage.lockFocus()
<
<       // set the fill color from the main window //
<       MainWindow.backgroundColor.setFill()
<
<       // fill the rectangle //
<       NSBezierPath(rect:NSMakeRect(0, 0, 200, 200)).fill()
<
<       // unlock the window color preview image //
<       WindowColorPreviewImage.unlockFocus()
4002,4003c3990,3991
<       // update the window color preview image //
<       WindowColorPreview.image = WindowColorPreviewImage
---
>       // update the window color pattern preview //
>       WindowColorPreview.image = WindowBackgroundPattern
4026,4027c4014,4015
<       // loop over the texture x index //
<       for TextureX in 0 ..< 10
---
>       // loop over the roughness x index //
>       for RoughnessXIndex in 0 ..< 10
4029,4030c4017,4018
<          // loop over the texture y index //
<          for TextureY in 0 ..< 10
---
>          // loop over the roughness y index //
>          for RoughnessYIndex in 0 ..< 10
4036,4037c4024,4025
<             NSBezierPath(rect:NSMakeRect(CGFloat(TextureX * 4), CGFloat(TextureY * 4) + 2.5, 4, 1))
<                         .fill()
---
>             NSBezierPath(rect:NSMakeRect(CGFloat(RoughnessXIndex * 4), CGFloat(RoughnessYIndex * 4)
>                         + 2.5, 4, 1)).fill()
4040,4041c4028,4029
<             NSBezierPath(rect:NSMakeRect(CGFloat(TextureX * 4) + 2.5, CGFloat(TextureY * 4), 1, 4))
<                         .fill()
---
>             NSBezierPath(rect:NSMakeRect(CGFloat(RoughnessXIndex * 4) + 2.5,
>                         CGFloat(RoughnessYIndex * 4), 1, 4)).fill()
4047,4048c4035,4036
<             NSBezierPath(rect:NSMakeRect(CGFloat(TextureX * 4), CGFloat(TextureY * 4) + 0.5, 4, 1))
<                         .fill()
---
>             NSBezierPath(rect:NSMakeRect(CGFloat(RoughnessXIndex * 4), CGFloat(RoughnessYIndex * 4)
>                         + 0.5, 4, 1)).fill()
4051,4052c4039,4040
<             NSBezierPath(rect:NSMakeRect(CGFloat(TextureX * 4) + 0.5, CGFloat(TextureY * 4), 1, 4))
<                         .fill()
---
>             NSBezierPath(rect:NSMakeRect(CGFloat(RoughnessXIndex * 4) + 0.5,
>                         CGFloat(RoughnessYIndex * 4), 1, 4)).fill()
4058c4046
<             if (TextureX + TextureY) % 2 == 0
---
>             if (RoughnessXIndex + RoughnessYIndex) % 2 == 0
4061,4062c4049,4050
<                NSBezierPath(rect:NSMakeRect(CGFloat(TextureX * 4) + 0.5, CGFloat(TextureY * 4) +
<                            2.5, 1, 1)).fill()
---
>                NSBezierPath(rect:NSMakeRect(CGFloat(RoughnessXIndex * 4) + 0.5,
>                            CGFloat(RoughnessYIndex * 4) + 2.5, 1, 1)).fill()
4068,4069c4056,4057
<                NSBezierPath(rect:NSMakeRect(CGFloat(TextureX * 4) + 2.5, CGFloat(TextureY * 4) +
<                            0.5, 1, 1)).fill()
---
>                NSBezierPath(rect:NSMakeRect(CGFloat(RoughnessXIndex * 4) + 2.5,
>                            CGFloat(RoughnessYIndex * 4) + 0.5, 1, 1)).fill()
4193,4199d4180
<    // open the greenchess forum webpage //
<    @IBAction func OpenGreenChessForumWebpage(_:AnyObject)
<    {
<       // open the greenchess forum webpage //
<       NSWorkspace.shared.open(URL(string:"https://greenchess.createaforum.com")!)
<    }
<
4207,4213d4187
<    // open the privacy policy webpage //
<    @IBAction func OpenPrivacyPolicyWebpage(_:AnyObject)
<    {
<       // open the privacy policy webpage //
<       NSWorkspace.shared.open(URL(string:"https://greenchess.com/privacy")!)
<    }
<
4438,4461c4412
<    // refresh the diagram layer //
<    func RefreshDiagramLayer()
<    {
<       // clear the diagram layer context //
<       DiagramLayer.context?.clear(ViewBounds)
<
<       // loop over all the diagram paths //
<       for Index in 0 ..< DiagramPaths.count
<       {
<          // set the stroke color for this path //
<          DiagramLayer.context?.setStrokeColor(DiagramPathColors[Index])
<
<          // add the path to the diagram layer context //
<          DiagramLayer.context?.addPath(DiagramPaths[Index])
<
<          // stroke the current path //
<          DiagramLayer.context?.strokePath()
<       }
<
<       // update the display //
<       displayIgnoringOpacity(bounds)
<    }
<
<    // undo the diagram path //
---
>    // undo the draw path //
4467,4468c4418,4419
<       // if there is a diagram path to undo //
<       if !DiagramPaths.isEmpty
---
>       // if there is a shape to undo //
>       if Shapes.count > 0
4470,4475c4421,4422
<          // remove the last diagram path //
<          DiagramPaths.removeLast()
<
<          // remove the last diagram color //
<          DiagramPathColors.removeLast()
<       }
---
>          // remove the shape layer from the super layer //
>          Shapes[Shapes.count - 1].removeFromSuperlayer()
4477,4478c4424,4426
<       // refresh the diagram layer //
<       RefreshDiagramLayer()
---
>          // remove most current shape from the shape list //
>          Shapes.removeLast()
>       }
4486,4491d4433
<
<       // remove all the diagram paths //
<       DiagramPaths.removeAll()
<
<       // remove all the diagram path colors //
<       DiagramPathColors.removeAll()
4493,4494c4435,4443
<       // refresh the diagram layer //
<       RefreshDiagramLayer()
---
>       // loop over all the shapes //
>       for Index in 0 ..< Shapes.count
>       {
>          // remove the shape from the super layer //
>          Shapes[Index].removeFromSuperlayer()
>       }
>
>       // remove all the shapes from the shape list //
>       Shapes.removeAll()
6690,6694c6639,6640
<       // clear the diagram paths //
<       DiagramPaths.removeAll()
<
<       // clear the diagram path colors //
<       DiagramPathColors.removeAll()
---
>       // clear the diagram shapes //
>       Shapes.removeAll()
6785,6790c6731,6782
<       // get the window background hue //
<       WindowBackgroundHue = PreferenceInt("WindowBackgroundHue", Default:63, Min:0, Max:
<                                           WindowBackgroundColorView.MaxHue())
<
<       // initialize the window background color view //
<       WindowBackgroundColorView.Initialize(WindowBackgroundHue)
---
>       // get the background color set //
>       BackgroundColorSet = PreferenceInt("BackgroundColorSet", Default:6, Min:1, Max:7)
>
>       // get the rainbow background color //
>       RainbowBackgroundColor = PreferenceInt("RainbowBackgroundColor", Default:78, Min:0, Max:
>                                              RainbowBackgroundColors.count - 1)
>
>       // store the rainbow background colors //
>       RainbowBackgroundView.StoreColorSet(RainbowBackgroundColors, ChosenColor:
>                                           RainbowBackgroundColor)
>
>       // get the blue background color //
>       BlueBackgroundColor = PreferenceInt("BlueBackgroundColor", Default:200, Min:0, Max:
>                                           BlueBackgroundColors.count - 1)
>
>       // store the blue background colors //
>       BlueBackgroundView.StoreColorSet(BlueBackgroundColors, ChosenColor:BlueBackgroundColor)
>
>       // get the brown background color //
>       BrownBackgroundColor = PreferenceInt("BrownBackgroundColor", Default:200, Min:0, Max:
>                                            BrownBackgroundColors.count - 1)
>
>       // store the brown background colors //
>       BrownBackgroundView.StoreColorSet(BrownBackgroundColors, ChosenColor:BrownBackgroundColor)
>
>       // get the green background color //
>       GreenBackgroundColor = PreferenceInt("GreenBackgroundColor", Default:200, Min:0, Max:
>                                            GreenBackgroundColors.count - 1)
>
>       // store the green background colors //
>       GreenBackgroundView.StoreColorSet(GreenBackgroundColors, ChosenColor:GreenBackgroundColor)
>
>       // get the teal background color //
>       TealBackgroundColor = PreferenceInt("TealBackgroundColor", Default:200, Min:0, Max:
>                                           TealBackgroundColors.count - 1)
>
>       // store the teal background colors //
>       TealBackgroundView.StoreColorSet(TealBackgroundColors, ChosenColor:TealBackgroundColor)
>
>       // get the yellow background color //
>       YellowBackgroundColor = PreferenceInt("YellowBackgroundColor", Default:200, Min:0, Max:
>                                             YellowBackgroundColors.count - 1)
>
>       // store the yellow background colors //
>       YellowBackgroundView.StoreColorSet(YellowBackgroundColors, ChosenColor:YellowBackgroundColor)
>
>       // get the gray background color //
>       GrayBackgroundColor = PreferenceInt("GrayBackgroundColor", Default:200, Min:0, Max:
>                                           GrayBackgroundColors.count - 1)
>
>       // store the gray background colors //
>       GrayBackgroundView.StoreColorSet(GrayBackgroundColors, ChosenColor:GrayBackgroundColor)
6802c6794
<       SetBackgroundColor()
---
>       SetBackgroundColor(BackgroundColorSet)
6804,6805c6796,6798
<       // retrieve the saved board label hue //
<       LabelHue = PreferenceInt("LabelHue", Default:41, Min:0, Max:BoardLabelsColorView.MaxHue())
---
>       // retrieve the saved label color //
>       LabelColor = PreferenceInt("LabelColor", Default:FontColors.count - 1, Min:0, Max:
>                                  FontColors.count - 1)
6808c6801
<       BoardLabelsColorView.Initialize(LabelHue)
---
>       BoardLabelsColorView.StoreColorSet(FontColors, ChosenColor:LabelColor)
6811c6804
<       InfoHue = PreferenceInt("InfoHue", Default:27, Min:0, Max:GameInfoColorView.MaxHue())
---
>       InfoColor = PreferenceInt("InfoColor", Default:36, Min:0, Max:FontColors.count - 1)
6814c6807
<       GameInfoColorView.Initialize(InfoHue)
---
>       GameInfoColorView.StoreColorSet(FontColors, ChosenColor:InfoColor)
6817c6810
<       MoveListHue = PreferenceInt("MoveListHue", Default:27, Min:0, Max:MoveListColorView.MaxHue())
---
>       MoveListColor = PreferenceInt("MoveListColor", Default:36, Min:0, Max:FontColors.count - 1)
6820c6813
<       MoveListColorView.Initialize(MoveListHue)
---
>       MoveListColorView.StoreColorSet(FontColors, ChosenColor:MoveListColor)
6823c6816
<       GameListHue = PreferenceInt("GameListHue", Default:27, Min:0, Max:GameListColorView.MaxHue())
---
>       GameListColor = PreferenceInt("GameListColor", Default:36, Min:0, Max:FontColors.count - 1)
6826c6819
<       GameListColorView.Initialize(GameListHue)
---
>       GameListColorView.StoreColorSet(FontColors, ChosenColor:GameListColor)
6829c6822,6823
<       GameClockHue = PreferenceInt("GameClockHue", Default:27, Min:0, Max:ClockColorView.MaxHue())
---
>       GameClockColor = PreferenceInt("GameClockColor", Default:4, Min:0, Max:GameClockColors
>                                      .count - 1)
6832c6826
<       ClockColorView.Initialize(GameClockHue)
---
>       ClockColorView.StoreColorSet(GameClockColors, ChosenColor:GameClockColor)
6835c6829
<       GameClock.UpdateClockColor(ClockColorView.GetCurrentHue())
---
>       GameClock.UpdateClockColor(GameClockColors[GameClockColor])
6838c6832
<       GameClockPreview.UpdateClockColor(ClockColorView.GetCurrentHue())
---
>       GameClockPreview.UpdateClockColor(GameClockColors[GameClockColor])
6841,6842c6835,6836
<       GameClockBackgroundHue = PreferenceInt("GameClockBackgroundHue", Default:82, Min:0, Max:
<                                              ClockBackgroundColorView.MaxHue())
---
>       GameClockBackgroundColor = PreferenceInt("GameClockBackgroundColor", Default:4, Min:0, Max:
>                                                GameClockColors.count - 1)
6845c6839
<       ClockBackgroundColorView.Initialize(GameClockBackgroundHue)
---
>       ClockBackgroundColorView.StoreColorSet(GameClockColors, ChosenColor:GameClockBackgroundColor)
6848c6842
<       GameClock.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
---
>       GameClock.UpdateClockBackgroundColor(GameClockColors[GameClockBackgroundColor])
6851c6845
<       GameClockPreview.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
---
>       GameClockPreview.UpdateClockBackgroundColor(GameClockColors[GameClockBackgroundColor])
6854c6848
<       ClockBackgroundVisibility = PreferenceInt("ClockBackgroundVisibility", Default:30, Min:0,
---
>       ClockBackgroundVisibility = PreferenceInt("ClockBackgroundVisibility", Default:10, Min:0,
7002a6997,6999
>       // update the board tint overlay opacity //
>       TintOverlayOpacity = TintColorOpacity / 2.0
>
7013,7016c7010
<       BoardRoughnessOpacity = CGFloat(BoardRoughness) / 100.0
<
<       // create the board roughness texture //
<       CreateBoardRoughnessTexture()
---
>       BoardRoughnessOpacity = 0.15 * CGFloat(BoardRoughness) / 100.0
7019c7013
<       BoardRoughnessView.stringValue = "Roughness opacity is \(Float(BoardRoughnessOpacity))"
---
>       BoardRoughnessView.stringValue = "Roughness opacity is \(Float(BoardRoughness) / 100.0)"
7034,7035c7028,7029
<       BoardTintHue = PreferenceInt("BoardTintHue", Default:30, Min:0, Max:BoardTintHues.count
<                                    - 1)
---
>       BoardTintColor = PreferenceInt("BoardTintColor", Default:0, Min:0, Max:BoardTintColors.count
>                                      - 1)
7038c7032
<       BoardTintColorSlider.intValue = Int32(BoardTintHue)
---
>       BoardTintColorSlider.intValue = Int32(BoardTintColor)
7041c7035
<       BoardTintColorView.stringValue = "Tint color is \(BoardTintHue)"
---
>       BoardTintColorView.stringValue = "Tint color is \(BoardTintColor)"
7043,7045c7037,7038
<       // retrieve the chess board border texture color //
<       BorderTextureColor = PreferenceInt("BorderTextureColor", Default:BorderColors.count - 1,
<                                          Min:0, Max:BorderColors.count - 1)
---
>       // retrieve the chess board border texture //
>       BorderTexture = PreferenceInt("BorderTexture", Default:100, Min:0, Max:100)
7048c7041
<       BorderTextureSlider.intValue = Int32(BorderTextureColor)
---
>       BorderTextureSlider.intValue = Int32(BorderTexture)
7051c7044
<       BorderTextureView.stringValue = "Border texture is \(BorderTextureColor)"
---
>       BorderTextureView.stringValue = "Border texture is \(BorderTexture)"
7088,7105d7080
<
<       // retrieve the movie size preference //
<       MovieSize = PreferenceInt("MovieSize", Default:100, Min:50, Max:100)
<
<       // update the movie size slider //
<       MovieSizeSlider.intValue = Int32(MovieSize)
<
<       // display the movie size //
```

```
>       // display the movie size //
>       DisplayMovieSize()
<
<       // retrieve the movie frames per second from the slider preference //
<       MovieFPSFromSlider = PreferenceInt("MovieFPSFromSlider", Default:10, Min:0, Max:100)
<
<       // update the movie frames per second slider //
<       MovieFPSSlider.intValue = Int32(MovieFPSFromSlider)
<
<       // display the movie frames per second //
<       DisplayMovieFPS()
7260c7235
<       PreferencesSelection = PreferenceInt("PreferencesSelection", Default:20, Min:20, Max:26)
---
>       PreferencesSelection = PreferenceInt("PreferencesSelection", Default:20, Min:20, Max:25)
7298,7303d7272
<       // if the preferences selection is 26 //
<       else if PreferencesSelection == 26
<       {
<           // set the movie preferences setting //
<           MoviePreferencesAction(self)
<       }
7704,7705c7673,7674
<       // set that not currently a valid diagram path //
<       ValidPath = false
---
>       // set that not currently a valid diagram shape //
>       ValidShape = false
8085,8106d8053
<   // create the board roughness texture //
<   func CreateBoardRoughnessTexture()
<   {
<       // create the board roughness texture image //
<       BoardRoughnessTexture = NSImage(size:NSSize(width:Scale * 4, height:Scale * 4))
<
<       // lock the board roughness texture focus //
<       BoardRoughnessTexture.lockFocus()
<
<       // set the stroke color to light gray //
<       NSColor(red:0.85, green:0.85, blue:0.85, alpha:BoardRoughnessOpacity).set()
<
<       // draw the roughness circle //
<       RoughnessCircle.stroke()
<
<       // unlock the light square roughness focus //
<       BoardRoughnessTexture.unlockFocus()
<
<       // get the board roughness texture color from the board roughness texture image //
<       BoardRoughnessTextureColor = NSColor(patternImage:BoardRoughnessTexture)
<   }
<
8110,8112c8057,8062
<       // create the roughness circle //
<       RoughnessCircle = NSBezierPath(ovalIn:NSMakeRect(Scale, Scale, Scale * 2, Scale * 2))
---
>
>       // initialize the light square roughness image //
>       SquareRoughnessLight = NSImage(size:NSSize(width:SquareSize, height:SquareSize))
>
>       // initialize the dark square roughness image //
>       SquareRoughnessDark = NSImage(size:NSSize(width:SquareSize, height:SquareSize))
8115a8066,8115
>       // create the board tint white image //
>       BoardTintWhiteImage = NSImage(size:NSSize(width:SquareSize, height:SquareSize))
>
>       // create the board tint black image //
>       BoardTintBlackImage = NSImage(size:NSSize(width:SquareSize, height:SquareSize))
>
>       // lock the light square roughness focus //
>       SquareRoughnessLight.lockFocus()
>
>       // loop over the roughness x index //
>       for RoughnessXIndex in 0 ..< 25
>       {
>           // loop over the roughness y index //
>           for RoughnessYIndex in 0 ..< 25
>           {
>               // if is an even square //
>               if (RoughnessXIndex + RoughnessYIndex) % 2 == 0
>               {
>                   // append a circle to the roughness square //
>                   RoughnessCircles.appendOval(in:NSMakeRect(CGFloat(RoughnessXIndex * 3),
>                                               CGFloat(RoughnessYIndex * 3), 3, 3)))
>               }
>           }
>       }
>
>       // set the stroke color to black //
>       NSColor.black.set()
>
>       // draw the roughness circles //
>       RoughnessCircles.stroke()
>
>       // unlock the light square roughness focus //
>       SquareRoughnessLight.unlockFocus()
>
>       // lock the dark square roughness focus //
>       SquareRoughnessDark.lockFocus()
>
>       // set the stroke color to white //
>       NSColor.white.set()
>
>       // draw the roughness circles //
>       RoughnessCircles.stroke()
>
>       // unlock the dark square roughness focus //
>       SquareRoughnessDark.unlockFocus()
>
>       // update the chess board roughness layer //
>       ImageToLayer(CreateChessBoardGrid(SquareRoughnessLight, Dark:SquareRoughnessDark), Layer:
>                    CGLayerBoardRoughness, Size:CGBoardSize[2] * Scale)
>
8145c8145,8173
<
---
>
>       // lock the board tint white image focus //
>       BoardTintWhiteImage.lockFocus()
>
>       // set the background color to white //
>       NSColor.white.setFill()
>
>       // fill rectangle with the background color //
>       NSBezierPath(rect:NSMakeRect(0, 0, CGFloat(SquareSize), CGFloat(SquareSize))).fill()
>
>       // unlock the board tint white image focus //
>       BoardTintWhiteImage.unlockFocus()
>
>       // lock the board tint black image focus //
>       BoardTintBlackImage.lockFocus()
>
>       // set the background color to black //
>       NSColor.black.setFill()
>
>       // fill the rectangle with the background color //
>       NSBezierPath(rect:NSMakeRect(0, 0, CGFloat(SquareSize), CGFloat(SquareSize))).fill()
>
>       // unlock the board tint black image focus //
>       BoardTintBlackImage.unlockFocus()
>
>       // update the tint overlay cg layer //
>       ImageToLayer(CreateChessBoardGrid(BoardTintWhiteImage, Dark:BoardTintBlackImage), Layer:
>                    CGLayerTintOverlay, Size:CGBoardSize[2] * Scale)
>
8380a8409,8414
>       // allocate the tint overlay cg layer //
>       CGLayerTintOverlay = CGLayerOfSize(CGBoardSize[2] * Scale)
>
>       // allocate the board roughness cg layer //
>       CGLayerBoardRoughness = CGLayerOfSize(CGBoardSize[2] * Scale)
>
8393,8401d8426
<       // allocate the diagram layer //
<       DiagramLayer = CGLayerOfSize(CGBoardSize[2] * Scale)
<
<       // set the diagram layer path width //
<       DiagramLayer.context?.setLineWidth(3 * Scale)
<
<       // set the diagram path line cap style to round //
<       DiagramLayer.context?.setLineCap(CGLineCap.round)
<
8456c8481
<       AppNameView.stringValue = "GreenChess 8.1.3"
---
>       AppNameView.stringValue = "GreenChess 8.1.0"
8673,8688c8698,8700
<       // loop over all the border colors //
<       for Index in 0 ... 100
<       {
<           // append a color to the chess board border colors //
<           BorderColors.append(NSColor(red:sin(CGFloat(Index) * CGFloat.pi / 200) / 3, green:
<                               sin(CGFloat(Index) * CGFloat.pi / 200 + CGFloat.pi / 4) / 3, blue:
<                               sin(CGFloat(Index) * CGFloat.pi / 200 + CGFloat.pi / 2) / 3, alpha:1))
<
<           // append a board tint hue //
<           BoardTintHues.append(NSColor(hue:CGFloat(Index) / 100.0, saturation:1, brightness:1,
<                               alpha:1))
<       }
<
<       // mark board labels color view as for board labels color //
<       BoardLabelsColorView.InLabelColors = true
---
>
>       // initialize color set index to 0 //
>       Index = 0
8690,8691c8702,8776
<       // mark clock color view as for the clock border colors //
<       ClockColorView.InGameClockColors = true
---
>       // loop over red color //
>       for Red in 0 ..< 6
>       {
>           // loop over green color //
>           for Green in 0 ..< 6
>           {
>               // loop over blue color //
>               for Blue in 0 ..< 6
>               {
>                   // the window color scaling value //
>                   ColorScaling = CGFloat(Index) / 215
>
>                   // append the rainbow background color //
>                   RainbowBackgroundColors.append(NSColor(red:0.7 + CGFloat(Red) / 5.0 * 0.3, green:
>                                           0.7 + CGFloat(Green) / 5.0 * 0.3, blue:0.7 +
>                                           CGFloat(Blue) / 5.0 * 0.3, alpha:1))
>
>                   // append the game clock colors //
>                   GameClockColors.append(NSColor(red:CGFloat(Red) / 5.0, green:CGFloat(Green) /
>                                           5.0, blue:CGFloat(Blue) / 5.0, alpha:1))
>
>                   // append the board tint colors //
>                   BoardTintColors.append(NSColor(red:CGFloat(Red) / 5.0, green:CGFloat(Green)
>                                           / 5.0, blue:CGFloat(Blue) / 5.0, alpha:1))
>
>                   // append the font color //
>                   FontColors.append(NSColor(red:CGFloat(Red) / 5.0 * 0.7, green:CGFloat(Green) /
>                                           5.0 * 0.7, blue:CGFloat(Blue) / 5.0 * 0.7, alpha:1))
>
>                   // append the blue background color //
>                   BlueBackgroundColors.append(NSColor(red:0, green:0.1 + ColorScaling * 0.7, blue:1,
>                                           alpha:1))
>
>                   // append the green background color //
>                   GreenBackgroundColors.append(NSColor(red:0.1, green:0.35 + ColorScaling * 0.5,
>                                           blue:0.3 + ColorScaling * 0.4, alpha:1))
>
>                   // append the yellow background color //
>                   YellowBackgroundColors.append(NSColor(red:1, green:0.95, blue:0.2 + ColorScaling *
>                                           0.65, alpha:1))
>
>                   // append the gray background color //
>                   GrayBackgroundColors.append(NSColor(red:0.25 + ColorScaling * 0.75, green:0.25 +
>                                           ColorScaling * 0.75, blue:0.25 + ColorScaling * 0.75,
>                                           alpha:1))
>
>                   // append the brown background color //
>                   BrownBackgroundColors.append(NSColor(red:0.45 + ColorScaling * 0.35, green:0.4 +
>                                           ColorScaling * 0.35, blue:0.35 + ColorScaling * 0.35,
>                                           alpha:1))
>
>                   // append the teal background color //
>                   TealBackgroundColors.append(NSColor(red:0, green:0.5 + ColorScaling * 0.4, blue:
>                                           0.5 + ColorScaling * 0.4, alpha:1))
>
>                   // increment the color set index //
>                   Index += 1
>               }
>           }
>       }
>
>       // loop over all the border colors //
>       for Index in 0 ... 100
>       {
>           // append a color to the chess board border colors //
>           BorderColors.append(NSColor(red:sin(CGFloat(Index) * CGFloat.pi / 200) / 2, green:
>                               sin(CGFloat(Index) * CGFloat.pi / 200 + CGFloat.pi / 4) / 2, blue:
>                               sin(CGFloat(Index) * CGFloat.pi / 200 + CGFloat.pi / 2) / 2, alpha:1))
>       }
>
>       // mark board labels color view as for board labels color //
>       BoardLabelsColorView.InLabelColors = true
>
>       // mark clock color view as for the clock border colors //
>       ClockColorView.InGameClockColors = true
8704c8789,8810
<
---
>
>       // set the rainbow background view color set to 1 //
>       RainbowBackgroundView.ColorSet = 1
>
>       // set the blue background view color set to 2 //
>       BlueBackgroundView.ColorSet = 2
>
>       // set the brown background view color set to 3 //
>       BrownBackgroundView.ColorSet = 3
>
>       // set the green background view color set to 4 //
>       GreenBackgroundView.ColorSet = 4
>
>       // set the teal background view color set to 5 //
>       TealBackgroundView.ColorSet = 5
>
>       // set the yellow background view color set to 6 //
>       YellowBackgroundView.ColorSet = 6
>
>       // set the gray background view color set to 7 //
>       GrayBackgroundView.ColorSet = 7
8716,8736d8821
<
<       // update the clock preferences layout //
<       ClockPreferencesLayout()
<
<       // update the speed preferences layout //
<       SpeedPreferencesLayout()
<
<       // update the stockfish preferences layout //
<       StockfishPreferencesLayout()
<
<       // update the movie preferences layout //
<       MoviePreferencesLayout()
<
<       // update the board preferences layout //
<       BoardPreferencesLayout()
<
<       // update the background preferences layout //
<       BackgroundPreferencesLayout()
<
<       // update the font preferences layout //
<       FontPreferencesLayout()
8811c8896
<       // composite the bottom of the board //
---
>       // update the board labels layer //
8828c8913
<           BorderColors[BorderTextureColor].withAlphaComponent(BorderOpacity).setFill()
---
>           BorderColors[BorderTexture].withAlphaComponent(BorderOpacity).setFill()
8857c8942
<                               .foregroundColor:BoardLabelsColorView.GetCurrentHue()]
---
>                               .foregroundColor:FontColors[LabelColor]]
8908,8909c8993,8994
<       // set the board tint hue fill //
<       BoardTintHues[BoardTintHue].withAlphaComponent(TintColorOpacity / 4.0).setFill()
---
>       // set the board tint color fill //
>       BoardTintColors[BoardTintColor].withAlphaComponent(TintColorOpacity).setFill()
8912a8998,9003
>
>       // set the current graphics context alpha to the tint overlay opacity //
>       NSGraphicsContext.current?.cgContext.setAlpha(TintOverlayOpacity)
>
>       // draw the board tint overlay //
>       NSGraphicsContext.current?.cgContext.draw(CGLayerTintOverlay, in:ViewBounds)
8914,8915c9005,9006
<       // set the board roughness texture color //
<       BoardRoughnessTextureColor.setFill()
---
>       // set the current graphics context alpha to the board roughness opacity //
>       NSGraphicsContext.current?.cgContext.setAlpha(BoardRoughnessOpacity)
8917,8918c9008,9009
<       // draw the rectangle with the board roughness texture //
<       NSBezierPath(rect:InnerViewBounds).fill()
---
>       // draw the board roughness layer //
>       NSGraphicsContext.current?.cgContext.draw(CGLayerBoardRoughness, in:ViewBounds)
8939,8940c9030,9031
<       // set the border overlay color to white //
<       NSColor(red:1, green:1, blue:1, alpha:0.1).set()
---
>       // set the border overlay color to dark gray //
>       NSColor(red:0.25, green:0.25, blue:0.25, alpha:1).set()
9267,9268c9358,9359
<       // if the x coordinate is less than 31 //
<       if MouseLocation.x < 31
---
>       // if the x coordinate is less than 30 //
>       if MouseLocation.x < 30
9270,9271c9361,9362
<           // reset the mouse location x coordinate to 31 //
<           MouseLocation.x = 31
---
>           // reset the mouse location x coordinate to 32 //
>           MouseLocation.x = 32
9274,9275c9365,9366
<       // if the y coordinate is less than 31 //
<       if MouseLocation.y < 31
---
>       // if the y coordinate is less than 30 //
>       if MouseLocation.y < 30
9277,9278c9368,9369
<           // reset the mouse location y coordinate to 31 //
<           MouseLocation.y = 31
---
>           // reset the mouse location y coordinate to 32 //
>           MouseLocation.y = 32
9281,9282c9372,9373
<       // if the x coordinate is greater than 629 //
<       if MouseLocation.x > 629
---
>       // if the x coordinate is greater than 630 //
>       if MouseLocation.x > 630
9284,9285c9375,9376
<           // reset the mouse location x coordinate to 629 //
<           MouseLocation.x = 629
---
>           // reset the mouse location x coordinate to 628 //
>           MouseLocation.x = 628
9288,9289c9379,9380
<       // if the y coordinate is greater than 629 //
<       if MouseLocation.y > 629
---
>       // if the y coordinate is greater than 630 //
>       if MouseLocation.y > 630
9291,9292c9382,9383
<           // reset the mouse location y coordinate to 629 //
<           MouseLocation.y = 629
---
>           // reset the mouse location y coordinate to 628 //
>           MouseLocation.y = 628
9354,9355c9445,9446
<       // set that not currently a valid diagram path //
<       ValidPath = false
---
>       // set that not currently a valid draw shape //
>       ValidShape = false
9359c9450,9458
<
---
>
>       // if the mouse location is not on the chess board //
>       if MouseLocation.x < 30 || MouseLocation.y < 30 || MouseLocation.x > 630 || MouseLocation.y >
>           630
>       {
>           // exit the function since mouse location is not within the chess board //
>           return
>       }
9363,9365c9462,9464
<       // set that the diagram path is valid //
<       ValidPath = true
<
---
>       // set that the current shape is valid //
>       ValidShape = true
>
9371,9385c9470,9475
<       // append a path to the diagram paths //
<       DiagramPaths.append(CGMutablePath())
<
<       // append a path color to the diagram path colors //
<       DiagramPathColors.append(DiagramColor.cgColor)
<
<       // set the diagram layer stroke color //
<       DiagramLayer.context?.setStrokeColor(DiagramColor.cgColor)
<
<       // set the path location //
<       PathLocation = CGPoint(x:MouseLocation.x * Scale, y:MouseLocation.y * Scale)
<
<       // move the diagram path to the first location //
<       DiagramPaths[DiagramPaths.count - 1].move(to:PathLocation)
---
>       // create a shape layer //
>       Shape = CAShapeLayer()
>
>       // create a new mutable shape path //
>       ShapePath = CGMutablePath()
9387,9391c9477,9502
<       // set the diagram path starting point //
<       DiagramLayer.context?.move(to:PathLocation)
<
<       // save the previous mouse location //
<       PreviousMouseLocation = MouseLocation
---
>       // set the shape line width //
>       Shape.lineWidth = 4
>
>       // set the shape line cap style to round //
>       Shape.lineCap = "kCALineCapRound"
>
>       // set the shape line join style to round //
>       Shape.lineJoin = "kCALineJoinRound"
>
>       // set the shape stroke color //
>       Shape.strokeColor = DiagramColor.cgColor
>
>       // set the shape fill color //
>       Shape.fillColor = NSColor.clear.cgColor
>
>       // set the shape path starting point //
>       ShapePath.move(to:CGPoint(x:MouseLocation.x, y:MouseLocation.y))
>
>       // assign the shape path to the shape //
>       Shape.path = ShapePath
>
>       // append a new shape layer //
>       Shapes.append(Shape)
>
>       // add the new shape layer to the view layer //
>       layer?.addSublayer(Shape)
9396,9403d9506
<
<       // if the mouse location is not on the inside chess board //
<       if MouseLocation.x < 30 || MouseLocation.y < 30 || MouseLocation.x > 630 || MouseLocation.y
<       > 630
<       {
<           // exit the function since mouse location is not within the chess board //
<           return
<       }
9559,9560c9662,9663
<       // if in diagramming mode and is a valid diagram path //
<       if Diagramming && ValidPath
---
>       // if in diagramming mode and is a valid shape //
>       if Diagramming && ValidShape
9562c9665
<           // keep the marker in bounds //
---
>           // keep the marker pen in bounds //
9564,9584c9667,9669
<
<           // set the path location //
<           PathLocation = CGPoint(x:MouseLocation.x * Scale, y:MouseLocation.y * Scale)
<
<           // add a line to the diagram path location //
<           DiagramPaths[DiagramPaths.count - 1].addLine(to:PathLocation)
<
<           // add a line to the diagram layer context path //
<           DiagramLayer.context?.addLine(to:PathLocation)
<
<           // stroke the diagram layer path //
<           DiagramLayer.context?.strokePath()
<
<           // display the area that was drawn in //
<           displayIgnoringOpacity(NSMakeRect(min(PreviousMouseLocation.x - 4, MouseLocation.x - 4),
<                                  min(PreviousMouseLocation.y - 4, MouseLocation.y - 4),
<                                  abs(PreviousMouseLocation.x - MouseLocation.x) + 8,
<                                  abs(PreviousMouseLocation.y - MouseLocation.y) + 8))
<
<           // save the previous mouse location //
<           PreviousMouseLocation = MouseLocation
---
>
>           // add the line to the shape path //
>           ShapePath.addLine(to:CGPoint(x:MouseLocation.x, y:MouseLocation.y))
9586,9588c9671,9674
<           // return since done diagramming this path //
---
>           // assign the shape path to the shape //
>           Shapes[Shapes.count - 1].path = ShapePath
>
>           // return since done with diagramming portion of mouse dragged //
9831,9832c9916,9917
<       // if diagramming and is a valid path //
<       if Diagramming && ValidPath
---
>       // if in diagramming mode and is a valid shape //
>       if Diagramming && ValidShape
9834c9919
<           // keep the marker in bounds //
---
>           // keep the marker pen in bounds //
9836,9844c9921,9929
<
<           // set the path location //
<           PathLocation = CGPoint(x:MouseLocation.x * Scale, y:MouseLocation.y * Scale)
<
<           // add a line to the path location for the diagram paths //
<           DiagramPaths[DiagramPaths.count - 1].addLine(to:PathLocation)
<
<           // add a line to the path location in the diagram layer context //
<           DiagramLayer.context?.addLine(to:PathLocation)
---
>
>           // add the line to the shape path //
>           ShapePath.addLine(to:CGPoint(x:MouseLocation.x, y:MouseLocation.y))
>
>           // assign the shape path to the shape //
>           Shapes[Shapes.count - 1].path = ShapePath
>
>           // set that no longer a valid shape //
>           ValidShape = false
9846,9855c9931
<           // stroke the current path in the diagram layer context //
<           DiagramLayer.context?.strokePath()
<
<           // refresh the diagram layer //
<           RefreshDiagramLayer()
<
<           // reset valid path to false //
<           ValidPath = false
<
<           // return since done diagramming this path //
---
>           // return since done with mouse up portion of diagramming mode //
10151,10157d10226
<
<       // if diagramming //
<       if Diagramming
<       {
<           // draw the diagram layer into the context //
<           NSGraphicsContext.current?.cgContext.draw(DiagramLayer, in:bounds)
<       }
10584,10586c10653,10654
<       // prepare to write a video of the given size //
<       PrepareToWriteAMovie(GameGalleryView.Width * CGFloat(MovieSize) / 100.0, VidHeight:
<                            GameGalleryView.Height * CGFloat(MovieSize) / 100.0)
---
>       // prepare to write a video of given size //
>       PrepareToWriteAMovie(GameGalleryView.Width, VidHeight:GameGalleryView.Height)
10598c10666
<           // append the pixel buffer with frame and specified duration //
---
>           // append the pixel buffer with frame duration of two seconds //
10602,10603c10670
<                       withPresentationTime:CMTimeMake(Int64(Float(FrameIndex * 10) / MovieFPS),
<                       10))
---
>                       withPresentationTime:CMTimeMake(Int64(FrameIndex * 2), 1))
10633c10700
<                       CMTimeMake(0, 10))
---
>                       CMTimeMake(0, 1))
10642c10709
<       // append the pixel buffer with specified duration //
---
>       // append the pixel buffer with duration of two seconds //
10644c10711
<                       CMTimeMake(Int64(50 + Float(FrameNumber * 10) / MovieFPS), 10))
---
>                       CMTimeMake(Int64(3 + FrameNumber * 2), 1))
10651,10652c10718
<       PrepareToWriteAMovie(frame.size.width * CGFloat(MovieSize) / 100.0, VidHeight:
<                            frame.size.height * CGFloat(MovieSize) / 100.0)
---
>       PrepareToWriteAMovie(frame.size.width, VidHeight:frame.size.height)
10677a10744,10749
>
>       // set the snapshot bitmap size //
>       SnapshotBitmap = bitmapImageRepForCachingDisplay(in:bounds)!
>
>       // draw the specified image into the bitmap representation //
>       cacheDisplay(in:bounds, to:SnapshotBitmap)
10702,10721c10774,10777
<       // draw the board with pieces layer into the context //
<       CGContexts[2].draw(CGLayersBoardWithPieces[2], in:ViewBounds)
<
<       // if diagramming //
<       if Diagramming
<       {
<           // draw the diagram layer into the context //
<           CGContexts[2].draw(DiagramLayer, in:ViewBounds)
<       }
<
<       // get the snapshot image representation from the context //
<       SnapshotImageRep = NSBitmapImageRep(cgImage:CGContexts[2].makeImage()!)
<
<       // set the snapshot image representation size //
<       SnapshotImageRep.size = bounds.size
<
<       // write the snapshot image //
<       try? SnapshotImageRep.representation(using:NSBitmapImageRep.FileType.jpeg, properties:[:])!
<                           .write(to:URL(fileURLWithPath:Filename), options:
<                           [.atomic])
---
>       // write the board snapshot //
>       try? NSBitmapImageRep(cgImage:SnapshotBitmap.cgImage!).representation(using:NSBitmapImageRep
>                           .FileType.jpeg, properties:[:])!.write(to:URL(fileURLWithPath:Filename),
>                           options:[.atomic])
11792,11794d11847
<
<       // clear the diagram layer context //
<       DiagramLayer.context?.clear(ViewBounds)
11796,11800c11849,11857
<       // remove all the diagram paths //
<       DiagramPaths.removeAll()
<
<       // remove all the diagram path colors //
<       DiagramPathColors.removeAll()
---
>       // loop over all the shapes //
>       for Index in 0 ..< Shapes.count
>       {
>           // remove the shape from the super layer //
>           Shapes[Index].removeFromSuperlayer()
>       }
>
>       // remove all the shapes from the shape list //
>       Shapes.removeAll()
12178,12179c12235,12236
<       // save the chosen label hue //
<       AppPreferences.set(LabelHue, forKey:"LabelHue")
---
>       // save the chosen label color //
>       AppPreferences.set(LabelColor, forKey:"LabelColor")
12181,12182c12238,12239
<       // save the chosen info hue //
<       AppPreferences.set(InfoHue, forKey:"InfoHue")
---
>       // save the chosen info color //
>       AppPreferences.set(InfoColor, forKey:"InfoColor")
12184,12185c12241,12242
<       // save the chosen move list hue //
<       AppPreferences.set(MoveListHue, forKey:"MoveListHue")
---
>       // save the chosen move list color //
>       AppPreferences.set(MoveListColor, forKey:"MoveListColor")
12187,12188c12244,12245
<       // save the chosen game list hue //
<       AppPreferences.set(GameListHue, forKey:"GameListHue")
---
>       // save the chosen game list color //
>       AppPreferences.set(GameListColor, forKey:"GameListColor")
12190,12191c12247,12248
<       // save the chosen window background hue //
<       AppPreferences.set(WindowBackgroundHue, forKey:"WindowBackgroundHue")
---
>       // save the chosen rainbow background color //
>       AppPreferences.set(RainbowBackgroundColor, forKey:"RainbowBackgroundColor")
12193,12194c12250,12251
<       // save the chosen game clock border hue //
<       AppPreferences.set(GameClockHue, forKey:"GameClockHue")
---
>       // save the chosen blue background color //
>       AppPreferences.set(BlueBackgroundColor, forKey:"BlueBackgroundColor")
12196,12197c12253,12266
<       // save the game clock background hue //
<       AppPreferences.set(GameClockBackgroundHue, forKey:"GameClockBackgroundHue")
---
>       // save the chosen brown background color //
>       AppPreferences.set(BrownBackgroundColor, forKey:"BrownBackgroundColor")
>
>       // save the chosen green background color //
>       AppPreferences.set(GreenBackgroundColor, forKey:"GreenBackgroundColor")
>
>       // save the chosen teal background color //
>       AppPreferences.set(TealBackgroundColor, forKey:"TealBackgroundColor")
>
>       // save the chosen game clock border color //
>       AppPreferences.set(GameClockColor, forKey:"GameClockColor")
>
>       // save the game clock background color //
>       AppPreferences.set(GameClockBackgroundColor, forKey:"GameClockBackgroundColor")
12201a12271,12279
>       // save the chosen yellow background color //
>       AppPreferences.set(YellowBackgroundColor, forKey:"YellowBackgroundColor")
>
>       // save the chosen gray background color //
>       AppPreferences.set(GrayBackgroundColor, forKey:"GrayBackgroundColor")
>
>       // save the chosen background color set //
>       AppPreferences.set(BackgroundColorSet, forKey:"BackgroundColorSet")
>
12209c12287
<       AppPreferences.set(BorderTextureColor, forKey:"BorderTextureColor")
---
>       AppPreferences.set(BorderTexture, forKey:"BorderTexture")
12226,12227c12304,12305
<       // save the user preference for the board tint hue //
<       AppPreferences.set(BoardTintHue, forKey:"BoardTintHue")
---
>       // save the user preference for the board tint color //
>       AppPreferences.set(BoardTintColor, forKey:"BoardTintColor")
12279,12284d12356
<
<       // update the user preference for movie frames per second from the slider //
<       AppPreferences.set(MovieFPSFromSlider, forKey:"MovieFPSFromSlider")
<
<       // update the user preference for the movie size //
<       AppPreferences.set(MovieSize, forKey:"MovieSize")
12559,12560c12631,12632
<   // update the chess piece set type //
<   @IBAction func UpdatePieceSetType(_:AnyObject)
---
>   // set the chess piece set type //
>   @IBAction func SetPieceSetType(_:AnyObject)
12562,12566c12634,12635
<       // if chess piece set type is different //
<       if PieceSetButton.indexOfSelectedItem != ChessPieceSetType
<       {
<           // get the chess piece set type from the piece set button //
<           ChessPieceSetType = PieceSetButton.indexOfSelectedItem
---
>       // get the chess piece set type from the piece set button //
>       ChessPieceSetType = PieceSetButton.indexOfSelectedItem
12568,12569c12637,12638
<           // update the board and captured pieces //
<           UpdateBoardAndCapturedPieces()
---
>       // update the board and captured pieces //
>       UpdateBoardAndCapturedPieces()
12571,12573c12640,12641
<           // update the position editor toolbar buttons //
<           UpdatePositionEditorToolbarButtons()
<       }
---
>       // update the position editor toolbar buttons //
>       UpdatePositionEditorToolbarButtons()
12579,12583c12647,12648
<       // if the board type is different //
<       if BoardTexturesButton.indexOfSelectedItem != BoardType
<       {
<           // get the board type from the board textures button //
<           BoardType = BoardTexturesButton.indexOfSelectedItem
---
>       // get the board type from the board textures button //
>       BoardType = BoardTexturesButton.indexOfSelectedItem
12585,12587c12650,12652
<           // convert the board type image to the board type cg layer //
<           ImageToLayer(ChessBoardImages[BoardType], Layer:CGLayerBoardType, Size:CGBoardSize[2] *
<                        Scale)
---
>       // convert the board type image to the board type cg layer //
>       ImageToLayer(ChessBoardImages[BoardType], Layer:CGLayerBoardType, Size:CGBoardSize[2] *
>                    Scale)
12589,12591c12654,12655
<           // recomposite the board for the updated board type layer //
<           RecompositeBoard(true)
<       }
---
>       // recomposite the board for the updated board type layer //
>       RecompositeBoard(true)
12676c12740
<   func SetBackgroundColor()
---
>   func SetBackgroundColor(_ ColorSet:Int)
12678,12679c12742,12830
<       // get the current window background color //
<       WindowBackgroundHue = WindowBackgroundColorView.CurrentHue
---
>       // set the current background color set //
>       BackgroundColorSet = ColorSet
>
>       // set the rainbow background view border to false //
>       RainbowBackgroundView.Bordered = false
>
>       // set the blue background view border to false //
>       BlueBackgroundView.Bordered = false
>
>       // set the brown background view border to false //
>       BrownBackgroundView.Bordered = false
>
>       // set the green background view border to false //
>       GreenBackgroundView.Bordered = false
>
>       // set the teal background view border to false //
>       TealBackgroundView.Bordered = false
>
>       // set the yellow background view border to false //
>       YellowBackgroundView.Bordered = false
>
>       // set the gray background view border to false //
>       GrayBackgroundView.Bordered = false
>
>       // if color set is for rainbow colors //
>       if BackgroundColorSet == 1
>       {
>           // get the current rainbow background color //
>           RainbowBackgroundColor = RainbowBackgroundView.CurrentColor
>
>           // set the background color //
>           SetWindowBackgroundColor(RainbowBackgroundColors[RainbowBackgroundColor])
>
>           // set that the rainbow background color view is bordered //
>           RainbowBackgroundView.Bordered = true
>       }
>       // if color set is for blue colors //
>       else if BackgroundColorSet == 2
>       {
>           // get the current blue background color //
>           BlueBackgroundColor = BlueBackgroundView.CurrentColor
>
>           // set the background color //
>           SetWindowBackgroundColor(BlueBackgroundColors[BlueBackgroundColor])
>
>           // set that the blue background color view is bordered //
>           BlueBackgroundView.Bordered = true
>       }
>       // if color set is for brown colors //
>       else if BackgroundColorSet == 3
>       {
>           // get the current brown background color //
>           BrownBackgroundColor = BrownBackgroundView.CurrentColor
>
>           // set the background color //
>           SetWindowBackgroundColor(BrownBackgroundColors[BrownBackgroundColor])
>
>           // set that the brown background color view is bordered //
>           BrownBackgroundView.Bordered = true
>       }
>       // if color set is for green colors //
>       else if BackgroundColorSet == 4
>       {
>           // get the current green background color //
>           GreenBackgroundColor = GreenBackgroundView.CurrentColor
>
>           // set the background color //
>           SetWindowBackgroundColor(GreenBackgroundColors[GreenBackgroundColor])
>
>           // set that the green background color view is bordered //
>           GreenBackgroundView.Bordered = true
>       }
>       // if color set is for teal colors //
>       else if BackgroundColorSet == 5
>       {
>           // get the current teal background color //
>           TealBackgroundColor = TealBackgroundView.CurrentColor
>
>           // set the background color //
>           SetWindowBackgroundColor(TealBackgroundColors[TealBackgroundColor])
>
>           // set that the teal background color view is bordered //
>           TealBackgroundView.Bordered = true
>       }
>       // if color set is for yellow colors //
>       else if BackgroundColorSet == 6
>       {
>           // get the current yellow background color //
>           YellowBackgroundColor = YellowBackgroundView.CurrentColor
12681,12682c12832,12870
<       // set the window background color //
<       SetWindowBackgroundColor(WindowBackgroundColorView.GetCurrentHue())
---
>
>           // set the background color //
>           SetWindowBackgroundColor(YellowBackgroundColors[YellowBackgroundColor])
>
>           // set that the yellow background color view is bordered //
>           YellowBackgroundView.Bordered = true
>       }
>       // if color set is for gray colors //
>       else if BackgroundColorSet == 7
>       {
>           // get the current gray background color //
>           GrayBackgroundColor = GrayBackgroundView.CurrentColor
>
>           // set the background color //
>           SetWindowBackgroundColor(GrayBackgroundColors[GrayBackgroundColor])
>
>           // set that the gray background color view is bordered //
>           GrayBackgroundView.Bordered = true
>       }
>
>       // refresh the rainbow background color view //
>       RainbowBackgroundView.displayIgnoringOpacity(RainbowBackgroundView.bounds)
>
>       // refresh the blue background color view //
>       BlueBackgroundView.displayIgnoringOpacity(BlueBackgroundView.bounds)
>
>       // refresh the brown background color view //
>       BrownBackgroundView.displayIgnoringOpacity(BrownBackgroundView.bounds)
>
>       // refresh the green background color view //
>       GreenBackgroundView.displayIgnoringOpacity(GreenBackgroundView.bounds)
>
>       // refresh the teal background color view //
>       TealBackgroundView.displayIgnoringOpacity(TealBackgroundView.bounds)
>
>       // refresh the yellow background color view //
>       YellowBackgroundView.displayIgnoringOpacity(YellowBackgroundView.bounds)
>
>       // refresh the gray background color view //
>       GrayBackgroundView.displayIgnoringOpacity(GrayBackgroundView.bounds)
12689c12877,12880
<       LabelHue = BoardLabelsColorView.CurrentHue
---
>       LabelColor = BoardLabelsColorView.CurrentColor
>
>       // refresh the board labels color view //
>       BoardLabelsColorView.displayIgnoringOpacity(BoardLabelsColorView.bounds)
12699,12700c12890,12894
<       InfoHue = InfoColorView.CurrentHue
```

```
<
---
>         InfoColor = GameInfoColorView.CurrentColor
>
>         // refresh the game info color view //
>         GameInfoColorView.displayIgnoringOpacity(GameInfoColorView.bounds)
>
12709c12903,12906
<         MoveListHue = MoveListColorView.CurrentHue
---
>         MoveListColor = MoveListColorView.CurrentColor
>
>         // refresh the move list color view //
>         MoveListColorView.displayIgnoringOpacity(MoveListColorView.bounds)
12719c12916,12919
<         GameClockBackgroundHue = ClockBackgroundColorView.CurrentHue
---
>         GameClockBackgroundColor = ClockBackgroundColorView.CurrentColor
>
>         // update the clock background color view //
>         ClockBackgroundColorView.displayIgnoringOpacity(ClockBackgroundColorView.bounds)
12722c12922
<         GameClock.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
---
>         GameClock.UpdateClockBackgroundColor(GameClockColors[GameClockBackgroundColor])
12725c12925
<         GameClockPreview.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
---
>         GameClockPreview.UpdateClockBackgroundColor(GameClockColors[GameClockBackgroundColor])
12735,12736c12935,12939
<         GameClockHue = ClockColorView.CurrentHue
<
---
>         GameClockColor = ClockColorView.CurrentColor
>
>         // update the clock color view //
>         ClockColorView.displayIgnoringOpacity(ClockColorView.bounds)
>
12738c12941
<         GameClock.UpdateClockColor(ClockColorView.GetCurrentHue())
---
>         GameClock.UpdateClockColor(GameClockColors[GameClockColor])
12741c12944
<         GameClockPreview.UpdateClockColor(ClockColorView.GetCurrentHue())
---
>         GameClockPreview.UpdateClockColor(GameClockColors[GameClockColor])
12751c12954,12957
<         GameListHue = GameListColorView.CurrentHue
---
>         GameListColor = GameListColorView.CurrentColor
>
>         // refresh the game list color view //
>         GameListColorView.displayIgnoringOpacity(GameListColorView.bounds)
12842,12843c13048,13076
<         // update the window colors background text field state //
<         WindowColorsBackgroundField.isHidden = State
---
>         // update the rainbow background text field state //
>         RainbowBackgroundField.isHidden = State
>
>         // update the rainbow background view state //
>         RainbowBackgroundView.isHidden = State
>
>         // update the blue background text field state //
>         BlueBackgroundField.isHidden = State
>
>         // update the blue background view state //
>         BlueBackgroundView.isHidden = State
>
>         // update the brown background text field state //
>         BrownBackgroundField.isHidden = State
>
>         // update the brown background view state //
>         BrownBackgroundView.isHidden = State
>
>         // update the green background text field state //
>         GreenBackgroundField.isHidden = State
>
>         // update the green background view state //
>         GreenBackgroundView.isHidden = State
>
>         // update the teal background text field state //
>         TealBackgroundField.isHidden = State
>
>         // update the teal background view state //
>         TealBackgroundView.isHidden = State
12845,12846c13078,13088
<         // update the window background color view state //
<         WindowBackgroundColorView.isHidden = State
---
>         // update the yellow background text field state //
>         YellowBackgroundField.isHidden = State
>
>         // update the yellow background view state //
>         YellowBackgroundView.isHidden = State
>
>         // update the gray background text field state //
>         GrayBackgroundField.isHidden = State
>
>         // update the gray background view state //
>         GrayBackgroundView.isHidden = State
12917a13160
>
13019a13263,13265
>         // update the window color preview frame //
>         WindowColorPreview.frame = NSMakeRect(310, 576, 40, 40)
>
13021c13267
<         WindowTextureOpacitySlider.frame = NSMakeRect(30, 585, 600, 13)
---
>         WindowTextureOpacitySlider.frame = NSMakeRect(30, 555, 600, 13)
13024c13270
<         WindowTextureOpacityView.frame = NSMakeRect(30, 567, 600, 20)
---
>         WindowTextureOpacityView.frame = NSMakeRect(30, 537, 600, 20)
13026,13027c13272,13273
<         // update the window background color text field frame //
<         WindowColorsBackgroundField.frame = NSMakeRect(230, 540, 200, 20)
---
>         // update the rainbow background text field frame //
>         RainbowBackgroundField.frame = NSMakeRect(230, 510, 200, 20)
13029,13033c13275,13312
<         // update the window background color view frame //
<         WindowBackgroundColorView.frame = NSMakeRect(30, 500, 600, 40)
<
<         // update the window color preview frame //
<         WindowColorPreview.frame = NSMakeRect(230, 230, 200, 200)
---
>         // update the rainbow background view frame //
>         RainbowBackgroundView.frame = NSMakeRect(30, 470, 600, 40)
>
>         // update the blue background text field frame //
>         BlueBackgroundField.frame = NSMakeRect(230, 435, 200, 20)
>
>         // update the blue background view frame //
>         BlueBackgroundView.frame = NSMakeRect(30, 395, 600, 40)
>
>         // update the brown background text field frame //
>         BrownBackgroundField.frame = NSMakeRect(230, 360, 200, 20)
>
>         // update the brown background view frame //
>         BrownBackgroundView.frame = NSMakeRect(30, 320, 600, 40)
>
>         // update the green background text field frame //
>         GreenBackgroundField.frame = NSMakeRect(230, 285, 200, 20)
>
>         // update the green background view frame //
>         GreenBackgroundView.frame = NSMakeRect(30, 245, 600, 40)
>
>         // update the teal background text field frame //
>         TealBackgroundField.frame = NSMakeRect(230, 210, 200, 20)
>
>         // update the teal background view frame //
>         TealBackgroundView.frame = NSMakeRect(30, 170, 600, 40)
>
>         // update the yellow background text field frame //
>         YellowBackgroundField.frame = NSMakeRect(230, 135, 200, 20)
>
>         // update the yellow background view frame //
>         YellowBackgroundView.frame = NSMakeRect(30, 95, 600, 40)
>
>         // update the gray background text field frame //
>         GrayBackgroundField.frame = NSMakeRect(230, 60, 200, 20)
>
>         // update the gray background view frame //
>         GrayBackgroundView.frame = NSMakeRect(30, 20, 600, 40)
13128,13143d13406
<     // the movie preferences layout //
<     func MoviePreferencesLayout()
<     {
<         // update the movie size slider frame //
<         MovieSizeSlider.frame = NSMakeRect(30, 585, 600, 13)
<
<         // update the movie size text view frame //
<         MovieSizeView.frame = NSMakeRect(30, 567, 600, 20)
<
<         // update the movie frames per second slider frame //
<         MovieFPSSlider.frame = NSMakeRect(30, 535, 600, 13)
<
<         // update the movie frames per second text view frame //
<         MovieFPSView.frame = NSMakeRect(30, 517, 600, 20)
<     }
<
13191,13206d13453
<     // the movie preferences state //
<     func MoviePreferencesToState(_ State:Bool)
<     {
<         // update the movie size slider state //
<         MovieSizeSlider.isHidden = State
<
<         // update the movie size view state //
<         MovieSizeView.isHidden = State
<
<         // update the movie frames per second slider state //
<         MovieFPSSlider.isHidden = State
<
<         // update the movie frames per second view state //
<         MovieFPSView.isHidden = State
<     }
<
13254,13257c13501,13502
<     // update the preference states //
<     func UpdatePreferenceStates(_ Preview:Bool, ClockState:Bool, SpeedState:Bool, StockfishState:
<                                  Bool, FontState:Bool, BackgroundState:Bool, BoardState:Bool,
<                                  MovieState:Bool)
---
>     // the clock preferences action //
>     @IBAction func ClockPreferencesAction(_:AnyObject)
13259,13260c13504,13505
<         // set whether to update the board preview in preferences //
<         UpdatePreview = Preview
---
>         // set that don't need to update the board preview //
>         UpdatePreview = false
13262,13268c13507,13509
<         // if need to update the board preview //
<         if UpdatePreview
<         {
<             // set that need to update the display //
<             displayIgnoringOpacity(bounds)
<         }
<
---
>         // update the game clock preview //
>         GameClockPreview.UpdateClock(0, ElapsedBlackTime:0)
>
13270c13511
<         ClockPreferencesToState(ClockState)
---
>         ClockPreferencesToState(false)
13273c13514
<         SpeedPreferencesToState(SpeedState)
---
>         SpeedPreferencesToState(true)
13276c13517
<         StockfishPreferencesToState(StockfishState)
---
>         StockfishPreferencesToState(true)
13279c13520
<         FontPreferencesToState(FontState)
---
>         FontPreferencesToState(true)
13282c13523
<         BackgroundPreferencesToState(BackgroundState)
---
>         BackgroundPreferencesToState(true)
13285,13295c13526
<         BoardPreferencesToState(BoardState)
<
<         // update the movie preferences state //
<         MoviePreferencesToState(MovieState)
<     }
<
<     // the clock preferences action //
<     @IBAction func ClockPreferencesAction(_:AnyObject)
<     {
<         // update the game clock preview //
<         GameClockPreview.UpdateClock(0, ElapsedBlackTime:0)
---
>         BoardPreferencesToState(true)
13297,13300c13528,13529
<         // update the preference states //
<         UpdatePreferenceStates(false, ClockState:false, SpeedState:true, StockfishState:true,
<                                FontState:true, BackgroundState:true, BoardState:true, MovieState:
<                                true)
---
>         // update the clock preferences layout //
>         ClockPreferencesLayout()
13306,13309c13535,13557
<         // update the preference states //
<         UpdatePreferenceStates(false, ClockState:true, SpeedState:false, StockfishState:true,
<                                FontState:true, BackgroundState:true, BoardState:true, MovieState:
<                                true)
---
>         // set that don't need to update the board preview //
>         UpdatePreview = false
>
>         // update the clock preferences state //
>         ClockPreferencesToState(true)
>
>         // update the speed preferences state //
>         SpeedPreferencesToState(false)
>
>         // update the stockfish preferences state //
>         StockfishPreferencesToState(true)
>
>         // update the font preferences state //
>         FontPreferencesToState(true)
>
>         // update the background preferences state //
>         BackgroundPreferencesToState(true)
>
>         // update the board preferences state //
>         BoardPreferencesToState(true)
>
>         // update the speed preferences layout //
>         SpeedPreferencesLayout()
13315,13326c13563,13585
<         // update the preference states //
<         UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:false,
<                                FontState:true, BackgroundState:true, BoardState:true, MovieState:true)
<     }
<
<     // the movie preferences action //
<     @IBAction func MoviePreferencesAction(_:AnyObject)
<     {
<         // update the preference states //
<         UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
<                                FontState:true, BackgroundState:true, BoardState:true, MovieState:
<                                false)
---
>         // set that don't need to update the board preview //
>         UpdatePreview = false
>
>         // update the clock preferences state //
>         ClockPreferencesToState(true)
>
>         // update the speed preferences state //
>         SpeedPreferencesToState(true)
>
>         // update the stockfish preferences state //
>         StockfishPreferencesToState(false)
>
>         // update the font preferences state //
>         FontPreferencesToState(true)
>
>         // update the background preferences state //
>         BackgroundPreferencesToState(true)
>
>         // update the board preferences state //
>         BoardPreferencesToState(true)
>
>         // update the stockfish prferences layout //
>         StockfishPreferencesLayout()
13332,13335c13591,13616
<         // update the preference states //
<         UpdatePreferenceStates(true, ClockState:true, SpeedState:true, StockfishState:true,
<                                FontState:true, BackgroundState:true, BoardState:false, MovieState:
<                                true)
---
>         // set that need to update the board preview //
>         UpdatePreview = true
>
>         // set that need to update the display //
>         displayIgnoringOpacity(bounds)
>
>         // update the clock preferences state //
>         ClockPreferencesToState(true)
>
>         // update the speed preferences state //
>         SpeedPreferencesToState(true)
>
>         // update the stockfish preferences state //
>         StockfishPreferencesToState(true)
>
>         // update the font preferences state //
>         FontPreferencesToState(true)
>
>         // update the background preferences state //
>         BackgroundPreferencesToState(true)
>
>         // update the board preferences state //
>         BoardPreferencesToState(false)
>
>         // update the board preferences layout //
>         BoardPreferencesLayout()
13341,13344c13622,13644
<         // update the preference states //
<         UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
<                                FontState:true, BackgroundState:false, BoardState:true, MovieState:
<                                true)
---
>         // set that don't need to update the board preview //
>         UpdatePreview = false
>
>         // update the clock preferences state //
>         ClockPreferencesToState(true)
>
>         // update the speed preferences state //
>         SpeedPreferencesToState(true)
>
>         // update the stockfish preferences state //
>         StockfishPreferencesToState(true)
>
>         // update the font preferences state //
>         FontPreferencesToState(true)
>
>         // update the background preferences state //
>         BackgroundPreferencesToState(false)
>
>         // update the board preferences state //
>         BoardPreferencesToState(true)
>
>         // update the background preferences layout //
>         BackgroundPreferencesLayout()
13350,13353c13650,13672
<         // update the preference states //
<         UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
<                                FontState:false, BackgroundState:true, BoardState:true, MovieState:
<                                true)
---
>         // set that don't need to update the board preview //
>         UpdatePreview = false
>
>         // update the clock preferences state //
>         ClockPreferencesToState(true)
>
>         // update the speed preferences state //
>         SpeedPreferencesToState(true)
>
>         // update the stockfish preferences state //
>         StockfishPreferencesToState(true)
>
>         // update the font preferences state //
>         FontPreferencesToState(false)
>
>         // update the background preferences state //
>         BackgroundPreferencesToState(true)
>
>         // update the board preferences state //
>         BoardPreferencesToState(true)
>
>         // update the font preferences layout //
>         FontPreferencesLayout()
14659,14663c14978,14979
<         // if the border texture slider is different //
<         if BorderTextureSlider.intValue != BorderTextureColor
<         {
<             // get the current border texture from the border texture slider //
<             BorderTextureColor = Int(BorderTextureSlider.intValue)
---
>         // get the current border texture from the border texture slider //
>         BorderTexture = Int(BorderTextureSlider.intValue)
14665,14666c14981,14982
<             // display the border texture for the chess board //
<             BorderTextureView.stringValue = "Border texture is \(BorderTextureColor)"
---
>         // display the border texture for the chess board //
>         BorderTextureView.stringValue = "Border texture is \(BorderTexture)"
14668,14670c14984,14985
<             // recomposite the board for the updated border layer //
<             RecompositeBoard(false)
<         }
---
>         // recomposite the board for the updated border layer //
>         RecompositeBoard(false)
14676,14680c14991,14992
<         // if clock background visibility is different //
<         if ClockBackgroundVisibilitySlider.intValue != ClockBackgroundVisibility
<         {
<             // update the clock background visibility //
<             ClockBackgroundVisibility = Int(ClockBackgroundVisibilitySlider.intValue)
---
>         // update the clock background visibility //
>         ClockBackgroundVisibility = Int(ClockBackgroundVisibilitySlider.intValue)
14682,14683c14994,14995
<             // display the clock background visibility //
<             DisplayClockBackgroundVisibility()
---
>         // display the clock background visibility //
>         DisplayClockBackgroundVisibility()
14685,14686c14997,14998
<             // update the game clock background visibility //
<             GameClock.UpdateClockBackgroundVisibility(ClockBackgroundVisibility)
---
>         // update the game clock background visibility //
>         GameClock.UpdateClockBackgroundVisibility(ClockBackgroundVisibility)
14688,14689c15000,15001
<             // update the game clock preview background visibility //
<             GameClockPreview.UpdateClockBackgroundVisibility(ClockBackgroundVisibility)
---
>         // update the game clock preview background visibility //
>         GameClockPreview.UpdateClockBackgroundVisibility(ClockBackgroundVisibility)
14691,14693c15003,15004
<             // update the game clock preview //
<             GameClockPreview.UpdateClock(0, ElapsedBlackTime:0)
<         }
---
>         // update the game clock preview //
>         GameClockPreview.UpdateClock(0, ElapsedBlackTime:0)
14699,14703c15010,15011
<         // if board roughness is different //
<         if BoardRoughnessSlider.intValue != BoardRoughness
<         {
<             // get the current roughness from the board roughness slider //
<             BoardRoughness = Int(BoardRoughnessSlider.intValue)
---
>         // get the current roughness from the board roughness slider //
>         BoardRoughness = Int(BoardRoughnessSlider.intValue)
14705,14709c15013,15014
<             // update the board roughness layer opacity //
<             BoardRoughnessOpacity = CGFloat(BoardRoughness) / 100.0
<
<             // create the board roughness texture //
<             CreateBoardRoughnessTexture()
---
>         // update the board roughness layer opacity //
>         BoardRoughnessOpacity = 0.15 * CGFloat(BoardRoughness) / 100.0
14711,14712c15016,15017
<             // update the board roughness view //
<             BoardRoughnessView.stringValue = "Roughness opacity is \(Float(BoardRoughnessOpacity))"
---
>         // update the board roughness view //
>         BoardRoughnessView.stringValue = "Roughness opacity is \(Float(BoardRoughness) / 100.0)"
14714,14716c15019,15020
<             // recomposite the board for the updated board roughness layer //
<             RecompositeBoard(true)
<         }
---
>         // recomposite the board for the updated board roughness layer //
>         RecompositeBoard(true)
14722,14726c15026,15027
<         // if the board tint color is different //
<         if BoardTintColorSlider.intValue != BoardTintHue
<         {
<             // get the board tint hue from the board tint color slider //
<             BoardTintHue = Int(BoardTintColorSlider.intValue)
---
>         // get the board tint color from the board tint color slider //
>         BoardTintColor = Int(BoardTintColorSlider.intValue)
14728,14729c15029,15030
<             // update the board tint hue //
<             BoardTintColorView.stringValue = "Tint color is \(BoardTintHue)"
---
>         // update the board tint color //
>         BoardTintColorView.stringValue = "Tint color is \(BoardTintColor)"
14731,14733c15032,15033
<             // recomposite the board for the updated board tint layer //
<             RecompositeBoard(true)
<         }
---
>         // recomposite the board for the updated board tint layer //
>         RecompositeBoard(true)
14739,14743c15039,15040
<         // if the square border is different //
<         if SquareBorderSlider.intValue != SquareBorder
<         {
<             // get the current square border from the square border slider //
<             SquareBorder = Int(SquareBorderSlider.intValue)
---
>         // get the current square border from the square border slider //
>         SquareBorder = Int(SquareBorderSlider.intValue)
14745,14746c15042,15043
<             // update the square border layer opacity //
<             SquareBorderOpacity = CGFloat(SquareBorder) / 200.0
---
>         // update the square border layer opacity //
>         SquareBorderOpacity = CGFloat(SquareBorder) / 200.0
14748,14749c15045,15046
<             // update the square border view //
<             SquareBorderView.stringValue = "Square border opacity is \(Float(SquareBorder) / 100.0)"
---
>         // update the square border view //
>         SquareBorderView.stringValue = "Square border opacity is \(Float(SquareBorder) / 100.0)"
14751,14753c15048,15049
<             // recomposite the board for the updated square border layer //
<             RecompositeBoard(true)
<         }
---
>         // recomposite the board for the updated square border layer //
>         RecompositeBoard(true)
14759,14763c15055,15056
<         // if the board tint visiblity is different //
<         if BoardTintVisibilitySlider.intValue != BoardTintVisibility
<         {
<             // get the board tint visibility from the board tint visibility slider //
<             BoardTintVisibility = Int(BoardTintVisibilitySlider.intValue)
---
>         // get the board tint visibility from the board tint visibility slider //
>         BoardTintVisibility = Int(BoardTintVisibilitySlider.intValue)
14765,14766c15058,15059
<             // update the board tint layer opacity //
<             TintColorOpacity = CGFloat(BoardTintVisibility) / 100.0
---
>         // update the board tint layer opacity //
>         TintColorOpacity = CGFloat(BoardTintVisibility) / 100.0
14768,14769c15061,15062
<             // update the board tint visibility view //
<             BoardTintVisibilityView.stringValue = "Tint opacity is \(TintColorOpacity)"
---
>         // update the board tint overlay opacity //
>         TintOverlayOpacity = TintColorOpacity / 2.0
14771,14773c15064,15068
<             // recomposite the board for the updated board tint and tint overlay layers //
<             RecompositeBoard(true)
<         }
---
>         // update the board tint visibility view //
>         BoardTintVisibilityView.stringValue = "Tint opacity is \(TintColorOpacity)"
>
>         // recomposite the board for the updated board tint and tint overlay layers //
>         RecompositeBoard(true)
14779,14783c15074,15075
<         // if the border visibility is different //
<         if BorderVisibilitySlider.intValue != BorderVisibility
<         {
<             // get the current border visibility from the border visibility slider //
<             BorderVisibility = Int(BorderVisibilitySlider.intValue)
---
>         // get the current border visibility from the border visibility slider //
>         BorderVisibility = Int(BorderVisibilitySlider.intValue)
14785,14786c15077,15078
<             // update the border opacity //
<             BorderOpacity = CGFloat(BorderVisibility) / 100.0
---
>         // update the border opacity //
>         BorderOpacity = CGFloat(BorderVisibility) / 100.0
14788,14789c15080,15081
<             // display the border visibility for the chess board //
<             BorderVisibilityView.stringValue = "Border opacity is \(BorderOpacity)"
---
>         // display the border visibility for the chess board //
>         BorderVisibilityView.stringValue = "Border opacity is \(BorderOpacity)"
14791,14793c15083,15084
<             // recomposite the board for the updated border and border overlay layers //
<             RecompositeBoard(false)
<         }
---
>         // recomposite the board for the updated border and border overlay layers //
>         RecompositeBoard(false)
14799,14803c15090,15091
<         // if the window texture opacity is different //
<         if WindowTextureOpacitySlider.intValue != WindowTextureOpacity
<         {
<             // get the current window texture opacity from the window texture opacity slider //
<             WindowTextureOpacity = Int(WindowTextureOpacitySlider.intValue)
---
>         // get the current window texture opacity from the window texture opacity slider //
>         WindowTextureOpacity = Int(WindowTextureOpacitySlider.intValue)
14805,14806c15093,15094
<             // display the window texture opacity //
<             DisplayWindowTextureOpacity()
---
>         // display the window texture opacity //
>         DisplayWindowTextureOpacity()
14808,14809c15096,15097
<             // set the window background color from the current window color //
<             SetWindowBackgroundColor(CurrentWindowColor)
---
>         // set the window background color from the current window color //
>         SetWindowBackgroundColor(CurrentWindowColor)
14811,14813c15099,15100
<             // recomposite the board for the updated color layer //
<             RecompositeBoard(false)
<         }
---
>         // recomposite the board for the updated color layer //
>         RecompositeBoard(false)
14819,14823c15106,15107
<         // if the animation rate is different //
<         if AnimationRateSlider.intValue != AnimationRate
<         {
<             // get the current board animation rate from the board animation rate slider //
<             AnimationRate = Int(AnimationRateSlider.intValue)
---
>         // get the current board animation rate from the board animation rate slider //
>         AnimationRate = Int(AnimationRateSlider.intValue)
14825,14827c15109,15110
<             // display the board animation rate //
<             DisplayAnimationRate()
<         }
---
>         // display the board animation rate //
>         DisplayAnimationRate()
14833,14837c15116,15117
<         // if the gallery animation rate is different //
<         if GalleryAnimationRateSlider.intValue != GalleryAnimationRate
<         {
<             // get the current gallery animation rate from the animation rate slider //
<             GalleryAnimationRate = Int(GalleryAnimationRateSlider.intValue)
---
>         // get the current gallery animation rate from the animation rate slider //
>         GalleryAnimationRate = Int(GalleryAnimationRateSlider.intValue)
14839,14840c15119,15120
<             // display the gallery animation rate //
<             DisplayGalleryAnimationRate()
---
>         // display the gallery animation rate //
>         DisplayGalleryAnimationRate()
14842,14844c15122,15123
<             // set the game gallery view animation rate //
<             GameGalleryView.SetAnimationRate(GalleryAnimationRate)
<         }
---
>         // set the game gallery view animation rate //
>         GameGalleryView.SetAnimationRate(GalleryAnimationRate)
14850,14854c15129,15130
<         // if the uci engine difficulty is different //
<         if UCIEngineDifficultySlider.intValue != UCIEngineDifficulty
<         {
<             // get the current uci engine difficulty from the uci engine difficulty slider //
<             UCIEngineDifficulty = Int(UCIEngineDifficultySlider.intValue)
---
>         // get the current uci engine difficulty from the uci engine difficulty slider //
>         UCIEngineDifficulty = Int(UCIEngineDifficultySlider.intValue)
14856,14858c15132,15133
<             // display the uci engine difficulty //
<             DisplayUCIEngineDifficulty()
<         }
---
>         // display the uci engine difficulty //
>         DisplayUCIEngineDifficulty()
14864,14868c15139,15140
<         // if the uci engine hash is different //
<         if UCIEngineHashSlider.intValue != UCIEngineHash
<         {
<             // get the current uci engine hash from the uci engine hash slider //
<             UCIEngineHash = Int(UCIEngineHashSlider.intValue)
---
>         // get the current uci engine hash from the uci engine hash slider //
>         UCIEngineHash = Int(UCIEngineHashSlider.intValue)
14870,14872c15142,15143
<             // display the uci engine hash //
<             DisplayUCIEngineHash()
<         }
---
>         // display the uci engine hash //
>         DisplayUCIEngineHash()
14878,14882c15149,15150
<         // if the uci engine riskiness is different //
<         if UCIEngineRiskinessSlider.intValue != UCIEngineRiskiness
<         {
<             // get the current uci engine riskiness from the uci engine riskiness slider //
<             UCIEngineRiskiness = Int(UCIEngineRiskinessSlider.intValue)
---
>         // get the current uci engine riskiness from the uci engine riskiness slider //
>         UCIEngineRiskiness = Int(UCIEngineRiskinessSlider.intValue)
14884,14886c15152,15153
<             // display the uci engine riskiness //
<             DisplayUCIEngineRiskiness()
<         }
---
>         // display the uci engine riskiness //
>         DisplayUCIEngineRiskiness()
14892,14896c15159,15160
<         // if the search time is different //
<         if SearchTimeSlider.intValue != SearchTime
<         {
<             // get the current game tree search time from the search time slider //
<             SearchTime = Int(SearchTimeSlider.intValue)
---
>         // get the current game tree search time from the search time slider //
>         SearchTime = Int(SearchTimeSlider.intValue)
14898,14945c15162,15163
<             // display the current search time //
<             DisplaySearchTime()
<         }
<     }
<
<     // update the movie frames per second //
<     @IBAction func UpdateMovieFPS(_:AnyObject)
<     {
<         // if the movie frames per second is different //
<         if MovieFPSSlider.intValue != MovieFPSFromSlider
<         {
<             // get the movie frames per second from slider value //
<             MovieFPSFromSlider = Int(MovieFPSSlider.intValue)
<
<             // display the movie frames per second //
<             DisplayMovieFPS()
<         }
<     }
<
<     // update the movie size //
<     @IBAction func UpdateMovieSize(_:AnyObject)
<     {
<         // if the movie size is different //
<         if MovieSizeSlider.intValue != MovieSize
<         {
<             // get the movie size from the movie size slider //
<             MovieSize = Int(MovieSizeSlider.intValue)
<
<             // display the movie size //
<             DisplayMovieSize()
<         }
<     }
<
<     // display the movie frames per second //
<     func DisplayMovieFPS()
<     {
<         // set the movie frames per second //
<         MovieFPS = round(Float(MovieFPSFromSlider) * 3.9 + 10) / 100.0
<
<         // set the movie frames per second view to show the frames per second //
<         MovieFPSView.stringValue = "Movie frames per second is \(MovieFPS)"
<     }
<
<     // display the movie size //
<     func DisplayMovieSize()
<     {
<         // set the movie size view to show the movie size //
<         MovieSizeView.stringValue = "Movie size is \(MovieSize)%"
---
>         // display the current search time //
>         DisplaySearchTime()
15698,15700c15916,15917
<         GameListData.RefreshGameList(GameListColorView.GetCurrentHue(), TheFont:
<                                      GetFont(GameListFontType, Size:CGFloat(14 + GameListSizes),
<                                      Bold:GameListBold))
---
>         GameListData.RefreshGameList(FontColors[GameListColor], TheFont:GetFont(GameListFontType,
>                                      Size:CGFloat(14 + GameListSizes), Bold:GameListBold))
15720c15937
<         GameInfoData.RefreshGameInfoList(GameInfoList, Color:GameInfoColorView.GetCurrentHue(), Font:
---
>         GameInfoData.RefreshGameInfoList(GameInfoList, Color:FontColors[InfoColor], Font:
15739c15956
<         PGNMovesData.RefreshPGNMovesList(PGNMovesList, Color:MoveListColorView.GetCurrentHue(), Font:
---
>         PGNMovesData.RefreshPGNMovesList(PGNMovesList, Color:FontColors[MoveListColor], Font:
17158c17375
<         AddFrame(0)
---
>         AddFrame(BoolToInt(RecordingMovie))
17195c17412
<             AddFrame(Index + 1)
---
>             AddFrame(Index + 1 + BoolToInt(RecordingMovie))
```