```
2c2
< // Swift source code of main.swift Version 8.2.8 at greenchess.com //
---
> // Swift source code of main.swift Version 8.2.7 at greenchess.com //
14,55d13
<     // the user preferences //
<     var Prefs:UserPreferences = UserPreferences()
<
<     // the gallery toolbar tabs //
<     var GalleryTabs:ToolbarTabs = ToolbarTabs()
<
<     // the game explorer toolbar tabs //
<     var GameExplorerTabs:ToolbarTabs = ToolbarTabs()
<
<     // the board preference controls //
<     var BoardControls:Controls = Controls()
<
<     // the list of board preference controls //
<     var BoardControlsList:[AnyObject] = []
<
<     // the game clock preference controls //
<     var ClockControls:Controls = Controls()
<
<     // the preference stockfish controls //
<     var StockfishControls:Controls = Controls()
<
<     // the preference font controls //
<     var FontControls:Controls = Controls()
<
<     // the list of font preference controls //
<     var FontControlsList:[AnyObject] = []
<
<     // the preference background controls //
<     var BackgroundControls:Controls = Controls()
<
<     // the preference speed controls //
<     var SpeedControls:Controls = Controls()
<
<     // the preference diagram controls //
<     var DiagramControls:Controls = Controls()
<
<     // the preference gallery controls //
<     var GalleryControls:Controls = Controls()
<
<     // the preference movie controls //
<     var MovieControls:Controls = Controls()
<
283c241,247
<
---
>
>     // whether font is bold or not //
>     var FontBoldString:String = ""
>
>     // the board label font size //
>     var BoardLabelsSize:CGFloat = 20
>
289a254,256
>     // the board labels attributes //
>     var BoardLabelsAttributes:NSDictionary = [:]
>
293,294c260,261
<     // the move list font //
<     var MoveListFont:FontsSet = FontsSet("MoveList")
---
>     // the board labels font type //
>     var BoardLabelsFontType:Int = 0
296,297c263,264
<     // the game info font //
<     var GameInfoFont:FontsSet = FontsSet("GameInfo")
---
>     // the move list font type //
>     var MoveListFontType:Int = 0
299,300c266,306
<     // the game list font //
<     var GameListFont:FontsSet = FontsSet("GameList")
---
>     // the game info font type //
>     var GameInfoFontType:Int = 0
>
>     // the game list font type //
>     var GameListFontType:Int = 0
>
>     // the board labels bold setting //
>     var BoardLabelsBold:Bool = false
>
>     // the move list bold setting //
>     var MoveListBold:Bool = false
>
>     // the game info bold setting //
>     var GameInfoBold:Bool = false
>
>     // the game list bold setting //
>     var GameListBold:Bool = false
>
>     // the board labels font sizes setting //
>     var BoardLabelsSizes:Int = 0
>
>     // the move list font sizes setting //
>     var MoveListSizes:Int = 0
>
>     // the game info font sizes setting //
>     var GameInfoSizes:Int = 0
>
>     // the game list sizes setting //
>     var GameListSizes:Int = 0
>
>     // the possible fonts //
>     var Fonts:[String] = ["American Typewriter", "Arial", "Arial Narrow", "Avenir Next",
>                           "Avenir Next Condensed", "Baskerville", "Bodoni 72", "Chalkboard",
>                           "Chalkboard SE", "Cochin", "Comic Sans MS", "Copperplate", "Courier",
>                           "Courier New", "Didot", "Georgia", "Gill Sans", "Helvetica",
>                           "Helvetica Neue", "Iowan Old Style", "Lucida Grande", "Menlo",
>                           "Noteworthy", "Optima", "Palatino", "PT Mono", "PT Serif", "Skia",
>                           "Snell Roundhand", "Times", "Times New Roman", "Trebuchet MS", "Verdana"]
>
>     // the board label font //
>     var BoardLabelsFont:NSFont = NSFont()
302,304d307
<     // the board labels font //
<     var BoardLabelsFont:FontsSet = FontsSet("BoardLabels")
<
322c325,337
<
---
```

```
>
>     // the current label hue index //
>     var LabelHue:Int = 0
>
>     // the current info hue index //
>     var InfoHue:Int = 0
>
>     // the current move list hue index //
>     var MoveListHue:Int = 0
>
>     // the current game list hue index //
>     var GameListHue:Int = 0
>
449,452d463
<     // the diagram toolbar marker colors dictionary //
<     let Colors:[String:Int] = ["303":0, "304":1, "305":2, "310":3, "309":4, "306":5, "307":6,
<                                "308":7]
<
860a872,877
>     // the play icon //
>     var PlayIcon:NSImage = NSImage(named:"start_icon")!
>
>     // the pause icon //
>     var PauseIcon:NSImage = NSImage(named:"pause_icon")!
>
992c1009,1012
<
---
>
>     // the app preferences //
>     var AppPreferences:UserDefaults = UserDefaults.standard
>
1092,1094d1111
<     // the about window toolbar //
<     @IBOutlet var AboutToolbar:NSToolbar!
<
1108a1126,1131
>
>     // the diagram blue color toolbar item //
>     @IBOutlet var DiagramBlueColorItem:NSToolbarItem!
>
>     // the diagram cyan color toolbar item //
>     @IBOutlet var DiagramCyanColorItem:NSToolbarItem!
1109a1133,1159
>     // the diagram magenta color toolbar item //
>     @IBOutlet var DiagramMagentaColorItem:NSToolbarItem!
>
>     // the diagram white color toolbar item //
>     @IBOutlet var DiagramWhiteColorItem:NSToolbarItem!
>
>     // the diagram yellow color toolbar item //
>     @IBOutlet var DiagramYellowColorItem:NSToolbarItem!
>
>     // the diagram black color toolbar item //
>     @IBOutlet var DiagramBlackColorItem:NSToolbarItem!
>
>     // the diagram green color toolbar item //
>     @IBOutlet var DiagramGreenColorItem:NSToolbarItem!
>
>     // the diagram red color toolbar item //
>     @IBOutlet var DiagramRedColorItem:NSToolbarItem!
>
>     // the diagram snapshot toolbar item //
>     @IBOutlet var DiagramSnapshotItem:NSToolbarItem!
>
>     // the undo path toolbar item //
>     @IBOutlet var UndoPathItem:NSToolbarItem!
>
>     // the clear diagram toolbar item //
>     @IBOutlet var ClearDiagramItem:NSToolbarItem!
>
1184c1234,1237
<
---
>
>     // button for if want to save the edited position from the position editor //
>     @IBOutlet var PositionEditorSaveToolbarButton:NSToolbarItem!
>
1190c1243,1285
<
---
>
>     // the position editor eraser button //
>     @IBOutlet var PositionEditorEraser:NSToolbarItem!
>
>     // the position editor white pawn //
>     @IBOutlet var PositionEditorWhitePawn:NSToolbarItem!
>
>     // the position editor white knight //
>     @IBOutlet var PositionEditorWhiteKnight:NSToolbarItem!
>
>     // the position editor white bishop //
>     @IBOutlet var PositionEditorWhiteBishop:NSToolbarItem!
>
>     // the position editor white rook //
>     @IBOutlet var PositionEditorWhiteRook:NSToolbarItem!
>
>     // the position editor white queen //
>     @IBOutlet var PositionEditorWhiteQueen:NSToolbarItem!
>
>     // the position editor white king //
>     @IBOutlet var PositionEditorWhiteKing:NSToolbarItem!
>
>     // the position editor black pawn //
>     @IBOutlet var PositionEditorBlackPawn:NSToolbarItem!
>
>     // the position editor black knight //
>     @IBOutlet var PositionEditorBlackKnight:NSToolbarItem!
>
>     // the position editor black bishop //
>     @IBOutlet var PositionEditorBlackBishop:NSToolbarItem!
>
>     // the position editor black rook //
>     @IBOutlet var PositionEditorBlackRook:NSToolbarItem!
>
>     // the position editor black queen //
>     @IBOutlet var PositionEditorBlackQueen:NSToolbarItem!
>
>     // the position editor black king //
>     @IBOutlet var PositionEditorBlackKing:NSToolbarItem!
>
>     // the exit diagram toolbar button //
>     @IBOutlet var ExitDiagramToolbarButton:NSToolbarItem!
>
1611c1706
<     let PromoteToDictionary:[Int:String] = [0:"q", 1:"r", 2:"b", 3:"n"]
---
```

```
>    let PromoteToDictionary:[String:String] = ["0":"q","1":"r", "2":"b", "3":"n"]
1627,1632d1721
<
<    // the preference controls dictionary //
<    var PrefControls:[Int:Controls] = [:]
<
<    // the previous preferences selection //
<    var PreviousPreferencesSelection:Int = 0
1634,1643d1722
<    // the about dictionary //
<    let AboutDictionary:[String:String] = ["greenchess":"greenchess.com", "support":
<                        "greenchess.com/contact", "forum":"greenchess.com/forum", "stockfish":
<                        "stockfishchess.org/download", "privacy":"greenchess.com/privacy",
<                        "license":"gnu.org/licenses/gpl-3.0.en.html"]
<
<    // the position pieces dictionary //
<    let PositionPieces:[String:Int] = ["204":1, "205":2, "206":3, "207":4, "208":5, "209":6,
<                                      "210":-1, "211":-2, "212":-3, "213":-4, "214":-5, "215":-6]
<
1771,1772c1850,1851
<    // the about window toolbar action //
<    @IBAction func AboutAction(_:AnyObject)
---
>    // open the greenchess webpage //
>    @IBAction func OpenGreenChessWebpage(_:AnyObject)
1774,1783c1853,1889
<       // if the dictionary item exists //
<       if AboutDictionary[AboutToolbar!.selectedItemIdentifier!.rawValue] != nil
<       {
<          // open the corresponding webpage //
<          NSWorkspace.shared.open(URL(string:"https://" + AboutDictionary[AboutToolbar!
<                                 .selectedItemIdentifier!.rawValue]!)!)
<
<          // deselect the selected toolbar item //
<          AboutToolbar.selectedItemIdentifier = nil
<       }
---
>       // open the greenchess webpage //
>       NSWorkspace.shared.open(URL(string:"https://greenchess.com")!)
>    }
>
>    // open the greenchess support webpage //
>    @IBAction func OpenGreenChessSupportWebpage(_:AnyObject)
>    {
>       // open the greenchess support webpage //
>       NSWorkspace.shared.open(URL(string:"https://greenchess.com/contact")!)
>    }
>
>    // open the greenchess forum webpage //
>    @IBAction func OpenGreenChessForumWebpage(_:AnyObject)
>    {
>       // open the greenchess forum webpage //
>       NSWorkspace.shared.open(URL(string:"https://greenchess.createaforum.com")!)
>    }
>
>    // open the stockfish support webpage //
>    @IBAction func OpenStockfishSupportWebpage(_:AnyObject)
>    {
>       // open the stockfish support webpage //
>       NSWorkspace.shared.open(URL(string:"https://stockfishchess.org/download")!)
>    }
>
>    // open the privacy policy webpage //
>    @IBAction func OpenPrivacyPolicyWebpage(_:AnyObject)
>    {
>       // open the privacy policy webpage //
>       NSWorkspace.shared.open(URL(string:"https://greenchess.com/privacy")!)
>    }
>
>    // open the gnu license webpage //
>    @IBAction func OpenGNULicenseWebpage(_:AnyObject)
>    {
>       // open the gnu license webpage //
>       NSWorkspace.shared.open(URL(string:"https://gnu.org/licenses/gpl-3.0.en.html")!)
1856,1858c1962,1968
<             // return the attributed string at the row index in the game list with white font //
<             TableCell.textField!.attributedStringValue = GameListData.SelectedRow(Row, GameListFont:
<                                                                                     GameListFont)
---
>             // return the attributed string at the row index in the game list //
>             TableCell.textField!.attributedStringValue = NSAttributedString(string:"\n" +
>                                                          GameListData.GameList[Row] + "\n",
>                                                          attributes:[NSAttributedString.Key
>                                                          .font:GameListData.Font,
>                                                          NSAttributedString.Key
>                                                          .foregroundColor:NSColor.white])
1864c1974
<             TableCell.textField!.attributedStringValue = GameListData.GetRow(Row)
---
>             TableCell.textField!.attributedStringValue = GameListData.AttributedGameList[Row]
2159c2269
<       GameExplorerTabs.UpdateState(true)
---
>       SetGameExplorerButtons(true)
2170a2281,2327
>
>    // set the game explorer buttons to the given state //
>    func SetGameExplorerButtons(_ State:Bool)
>    {
>       // if the state is true //
>       if State
>       {
>          // set the game review button to the play icon //
>          GameReviewButton.image = PlayIcon
>       }
>       // the state is false //
>       else
>       {
>          // set the game review button to the pause icon //
>          GameReviewButton.image = PauseIcon
>       }
>
>       // set the game review button to the given state //
>       GameReviewButton.isEnabled = State
>
>       // set the enter analysis mode button to the given state //
>       EnterAnalysisModeButton.isEnabled = State
>
>       // set the enter position editor mode button to the given state //
>       EnterPositionEditorModeButton.isEnabled = State
```

```
>
>       // set the enter diagram mode button to the given state //
>       EnterDiagramModeButton.isEnabled = State
>
>       // set the game explorer snapshot button to the given state //
>       GameExplorerSnapshotButton.isEnabled = State
>
>       // set the game explorer game to movie button to the given state //
>       GameExplorerMovieButton.isEnabled = State
>
>       // set the exit game explorer button to the given state //
>       ExitGameExplorerButton.isEnabled = State
>
>       // set the game explorer slider to the given state //
>       GameExplorerSlider.isEnabled = State
>
>       // set the game explorer left button to the given state //
>       GameExplorerLeft.isEnabled = State
>
>       // set the game explorer right button to the given state //
>       GameExplorerRight.isEnabled = State
>    }
2260,2261c2417,2418
<       AnimStart = CGPoint(x:abs(BoolTo().CG(RotateBoard) * 7 - StartSquare.XF) * Dims
<                       .GetSquareSize(2) + SquareCenter, y:abs(BoolTo().CG(RotateBoard) * 7 -
---
>       AnimStart = CGPoint(x:abs(BoolToCGFloat(RotateBoard) * 7 - StartSquare.XF) * Dims
>                       .GetSquareSize(2) + SquareCenter, y:abs(BoolToCGFloat(RotateBoard) * 7 -
2265,2266c2422,2423
<       AnimEnd = CGPoint(x:abs(BoolTo().CG(RotateBoard) * 7 - EndSquare.XF) * Dims
<                      .GetSquareSize(2) + SquareCenter, y:abs(BoolTo().CG(RotateBoard) * 7 -
---
>       AnimEnd = CGPoint(x:abs(BoolToCGFloat(RotateBoard) * 7 - EndSquare.XF) * Dims
>                      .GetSquareSize(2) + SquareCenter, y:abs(BoolToCGFloat(RotateBoard) * 7 -
2546c2703
<       HideInfoMenuItem.state = NSControl.StateValue(rawValue:BoolTo().Num(HideInfoPanel))
---
>       HideInfoMenuItem.state = NSControl.StateValue(rawValue:BoolToInt(HideInfoPanel))
2549c2706
<       LargeInfoMenuItem.state = NSControl.StateValue(rawValue:BoolTo().Num(LargeInfoPanel))
---
>       LargeInfoMenuItem.state = NSControl.StateValue(rawValue:BoolToInt(LargeInfoPanel))
2552c2709
<       SmallInfoMenuItem.state = NSControl.StateValue(rawValue:BoolTo().Num(SmallInfoPanel))
---
>       SmallInfoMenuItem.state = NSControl.StateValue(rawValue:BoolToInt(SmallInfoPanel))
2555c2712
<       Prefs.Save(HideInfoPanel, Key:"HideInfoPanel")
---
>       AppPreferences.set(HideInfoPanel, forKey:"HideInfoPanel")
2558c2715
<       Prefs.Save(SmallInfoPanel, Key:"SmallInfoPanel")
---
>       AppPreferences.set(SmallInfoPanel, forKey:"SmallInfoPanel")
2561c2718
<       Prefs.Save(LargeInfoPanel, Key:"LargeInfoPanel")
---
>       AppPreferences.set(LargeInfoPanel, forKey:"LargeInfoPanel")
2567,2569c2724,2726
<       WindowMaxSize = CGSize(width:CGFloat(660 + BoolTo().Num(!HideInfoPanel) *
<                            BoolTo().Num(SmallInfoPanel) * 340 + BoolTo().Num(!HideInfoPanel) *
<                            BoolTo().Num(LargeInfoPanel) * 600), height:660)
---
>       WindowMaxSize = CGSize(width:CGFloat(660 + BoolToInt(!HideInfoPanel) *
>                            BoolToInt(SmallInfoPanel) * 340 + BoolToInt(!HideInfoPanel) *
>                            BoolToInt(LargeInfoPanel) * 600), height:660)
2591c2748
<       HideClock = CGFloat(BoolTo().Num(HideGameClockInPanel))
---
>       HideClock = CGFloat(BoolToInt(HideGameClockInPanel))
3829a3987,4066
>    // get the user saved preference for the specified key //
>    func PreferenceInt(_ Key:String, Default:Int, Min:Int, Max:Int) -> Int
>    {
>       // if the user preference already exists //
>       if AppPreferences.bool(forKey:Key + "Exists")
>       {
>          // if the user preference is outside the valid range //
>          if (AppPreferences.integer(forKey:Key) < Min) || (AppPreferences.integer(forKey:Key) >
>             Max)
>          {
>             // return the default value //
>             return Default
>          }
>          // the user preference was within the valid range //
>          else
>          {
>             // return the user perference for this key //
>             return AppPreferences.integer(forKey:Key)
>          }
>       }
>       // the user preference doesn't exist yet //
>       else
>       {
>          // set that the user preference exists now //
>          AppPreferences.set(true, forKey:Key + "Exists")
>
>          // set the default value for the specified key //
>          AppPreferences.set(Default, forKey:Key)
>
>          // return the default value for the key //
>          return Default
>       }
>    }
>
>    // get the user saved preference for the key //
>    func PreferenceBool(_ Key:String, Default:Bool) -> Bool
>    {
>       // if the user preference exists //
>       if AppPreferences.bool(forKey:Key + "Exists")
>       {
>          // return the saved user preference //
>          return AppPreferences.bool(forKey:Key)
>       }
>       // the user preference does not yet exist //
>       else
>       {
>          // set that the user preference exists now //
>          AppPreferences.set(true, forKey:Key + "Exists")
>
>          // set the default value for the specified key //
>          AppPreferences.set(Default, forKey:Key)
>
>          // return the default value for this key //
>          return Default
>       }
>    }
>
```

```
>    // get the user saved preference for this key //
>    func PreferenceString(_ Key:String, Default:String) -> String
>    {
>       // if the user saved preference exists //
>       if AppPreferences.bool(forKey:Key + "Exists")
>       {
>          // return the user saved preference for this key //
>          return AppPreferences.string(forKey:Key)!
>       }
>       // the user saved preference does not yet exist //
>       else
>       {
>          // set that the user preference now exists for this key //
>          AppPreferences.set(true, forKey:Key + "Exists")
>
>          // set the default value for this key //
>          AppPreferences.set(Default, forKey:Key)
>
>          // return the default value for this key //
>          return Default
>       }
>    }
>
3834,3835c4071,4072
<       WindowBackgroundHue = Prefs.PreferenceInt("WindowBackgroundHue", Default:64, Min:0, Max:
<                                                WindowBackgroundColorView.MaxHue())
---
>       WindowBackgroundHue = PreferenceInt("WindowBackgroundHue", Default:64, Min:0, Max:
>                                          WindowBackgroundColorView.MaxHue())
3841c4078
<       WindowTextureOpacity = Prefs.PreferenceInt("WindowTextureOpacity", Default:0, Min:0, Max:100)
---
>       WindowTextureOpacity = PreferenceInt("WindowTextureOpacity", Default:0, Min:0, Max:100)
3851,3867c4088,4112
<
<       // read in the move list font //
<       MoveListFont.Read(MoveListColorView, DefaultHue:27, TheFont:MoveListFontButton, Sizes:
<                        MoveListSizesButton, Bold:MoveListBoldButton, TheScale:1)
<
<       // read in the game info font //
<       GameInfoFont.Read(GameInfoColorView, DefaultHue:27, TheFont:GameInfoFontButton, Sizes:
<                        GameInfoSizesButton, Bold:GameInfoBoldButton, TheScale:1)
<
<       // read in the game list font //
<       GameListFont.Read(GameListColorView, DefaultHue:27, TheFont:GameListFontButton, Sizes:
<                        GameListSizesButton, Bold:GameListBoldButton, TheScale:1)
<
<       // read in the board labels font //
<       BoardLabelsFont.Read(BoardLabelsColorView, DefaultHue:41, TheFont:BoardLabelsFontButton,
<                           Sizes:BoardLabelsSizesButton, Bold:BoardLabelsBoldButton, TheScale:2)
<
---
>
>       // retrieve the saved board label hue //
>       LabelHue = PreferenceInt("LabelHue", Default:41, Min:0, Max:BoardLabelsColorView.MaxHue())
>
>       // update the board labels color view //
>       BoardLabelsColorView.Initialize(LabelHue)
>
>       // retrieve the saved info color //
>       InfoHue = PreferenceInt("InfoHue", Default:27, Min:0, Max:GameInfoColorView.MaxHue())
>
>       // update the game info color view //
>       GameInfoColorView.Initialize(InfoHue)
>
>       // retrieve the saved move list color //
>       MoveListHue = PreferenceInt("MoveListHue", Default:27, Min:0, Max:MoveListColorView.MaxHue())
>
>       // update the move list color view //
>       MoveListColorView.Initialize(MoveListHue)
>
>       // retrieve the saved game list color //
>       GameListHue = PreferenceInt("GameListHue", Default:27, Min:0, Max:GameListColorView.MaxHue())
>
>       // update the game list color view //
>       GameListColorView.Initialize(GameListHue)
>
3869,3870c4114
<       GameClockHue = Prefs.PreferenceInt("GameClockHue", Default:27, Min:0, Max:ClockColorView
<                                         .MaxHue())
---
>       GameClockHue = PreferenceInt("GameClockHue", Default:27, Min:0, Max:ClockColorView.MaxHue())
3885,3886c4129,4130
<       GameClockBackgroundHue = Prefs.PreferenceInt("GameClockBackgroundHue", Default:82, Min:0, Max:
<                                                   ClockBackgroundColorView.MaxHue())
---
>       GameClockBackgroundHue = PreferenceInt("GameClockBackgroundHue", Default:82, Min:0, Max:
>                                             ClockBackgroundColorView.MaxHue())
3901,3902c4145,4146
<       ClockBackgroundVisibility = Prefs.PreferenceInt("ClockBackgroundVisibility", Default:30,
<                                                      Min:0, Max:100)
---
>       ClockBackgroundVisibility = PreferenceInt("ClockBackgroundVisibility", Default:30, Min:0,
>                                                Max:100)
3920c4164
<       ChessPieceSetType = Prefs.PreferenceInt("ChessPieceSetType", Default:0, Min:0, Max:3)
---
>       ChessPieceSetType = PreferenceInt("ChessPieceSetType", Default:0, Min:0, Max:3)
3929c4173
<       BoardType = Prefs.PreferenceInt("BoardType", Default:0, Min:0, Max:18)
---
>       BoardType = PreferenceInt("BoardType", Default:0, Min:0, Max:18)
3933a4178,4238
>       // retrieve the board labels font type //
>       BoardLabelsFontType = PreferenceInt("BoardLabelsFontType", Default:17, Min:0, Max:Fonts.count
>                                          - 1)
>
>       // set the current board labels font type //
>       BoardLabelsFontButton.selectItem(at:BoardLabelsFontType)
>
>       // retrieve the move list font type //
>       MoveListFontType = PreferenceInt("MoveListFontType", Default:17, Min:0, Max:Fonts.count - 1)
>
>       // set the current move list font type //
>       MoveListFontButton.selectItem(at:MoveListFontType)
>
>       // retrieve the game info font type //
>       GameInfoFontType = PreferenceInt("GameInfoFontType", Default:17, Min:0, Max:Fonts.count - 1)
>
>       // set the current game info font type //
>       GameInfoFontButton.selectItem(at:GameInfoFontType)
>
>       // retrieve the game list font type //
>       GameListFontType = PreferenceInt("GameListFontType", Default:17, Min:0, Max:Fonts.count - 1)
>
>       // set the current game list font type //
>       GameListFontButton.selectItem(at:GameListFontType)
>
>       // retrieve the board labels bold setting //
>       BoardLabelsBold = PreferenceBool("BoardLabelsBold", Default:false)
>
```

```
>        // set the current board labels bold setting //
>        BoardLabelsBoldButton.selectItem(at:BoolToInt(BoardLabelsBold))
>
>        // retrieve the move list bold setting //
>        MoveListBold = PreferenceBool("MoveListBold", Default:false)
>
>        // set the current move list bold setting //
>        MoveListBoldButton.selectItem(at:BoolToInt(MoveListBold))
>
>        // retrieve the game info bold setting //
>        GameInfoBold = PreferenceBool("GameInfoBold", Default:false)
>
>        // set the current game info bold setting //
>        GameInfoBoldButton.selectItem(at:BoolToInt(GameInfoBold))
>
>        // retrieve the game list bold setting //
>        GameListBold = PreferenceBool("GameListBold", Default:false)
>
>        // set the current game list bold setting //
>        GameListBoldButton.selectItem(at:BoolToInt(GameListBold))
>
>        // retrieve the board labels font sizes setting //
>        BoardLabelsSizes = PreferenceInt("BoardLabelsSizes", Default:2, Min:0, Max:2)
>
>        // set the current board labels font sizes setting //
>        BoardLabelsSizesButton.selectItem(at:BoardLabelsSizes)
>
>        // retrieve the move list font sizes setting //
>        MoveListSizes = PreferenceInt("MoveListSizes", Default:1, Min:0, Max:2)
>
>        // set the current move list font sizes setting //
>        MoveListSizesButton.selectItem(at:MoveListSizes)
>
3935c4240
<        NumberOfGridRows = Prefs.PreferenceInt("NumberOfGridRows", Default:3, Min:2, Max:4)
---
>        NumberOfGridRows = PreferenceInt("NumberOfGridRows", Default:3, Min:2, Max:4)
3939a4245,4256
>        // retrieve the game info sizes setting //
>        GameInfoSizes = PreferenceInt("GameInfoSizes", Default:1, Min:0, Max:2)
>
>        // set the current game info sizes setting //
>        GameInfoSizesButton.selectItem(at:GameInfoSizes)
>
>        // retrieve the game list sizes setting //
>        GameListSizes = PreferenceInt("GameListSizes", Default:1, Min:0, Max:2)
>
>        // set the current game list sizes setting //
>        GameListSizesButton.selectItem(at:GameListSizes)
>
3941c4258
<        AnimationRate = Prefs.PreferenceInt("AnimationRate", Default:75, Min:1, Max:100)
---
>        AnimationRate = PreferenceInt("AnimationRate", Default:75, Min:1, Max:100)
3950c4267
<        GalleryAnimationRate = Prefs.PreferenceInt("GalleryAnimationRate", Default:75, Min:1, Max:100)
---
>        GalleryAnimationRate = PreferenceInt("GalleryAnimationRate", Default:75, Min:1, Max:100)
3959c4276
<        IllustrationSize = Prefs.PreferenceInt("IllustrationSize", Default:200, Min:100, Max:250)
---
>        IllustrationSize = PreferenceInt("IllustrationSize", Default:200, Min:100, Max:250)
3968c4285
<        BorderVisibility = Prefs.PreferenceInt("BorderVisibility", Default:100, Min:0, Max:100)
---
>        BorderVisibility = PreferenceInt("BorderVisibility", Default:100, Min:0, Max:100)
3977c4294
<        BoardTintVisibility = Prefs.PreferenceInt("BoardTintVisibility", Default:0, Min:0, Max:100)
---
>        BoardTintVisibility = PreferenceInt("BoardTintVisibility", Default:0, Min:0, Max:100)
3986c4303
<        BoardRoughness = Prefs.PreferenceInt("BoardRoughness", Default:0, Min:0, Max:100)
---
>        BoardRoughness = PreferenceInt("BoardRoughness", Default:0, Min:0, Max:100)
3995c4312
<        SquareBorder = Prefs.PreferenceInt("SquareBorder", Default:0, Min:0, Max:100)
---
>        SquareBorder = PreferenceInt("SquareBorder", Default:0, Min:0, Max:100)
4007,4008c4324
<        BoardTintHue = Prefs.PreferenceInt("BoardTintHue", Default:30, Min:0, Max:BoardTintHues.count
<                                           - 1)
---
>        BoardTintHue = PreferenceInt("BoardTintHue", Default:30, Min:0, Max:BoardTintHues.count - 1)
4017,4018c4333,4334
<        BorderTextureColor = Prefs.PreferenceInt("BorderTextureColor", Default:BorderColors.count - 1,
<                                                 Min:0, Max:BorderColors.count - 1)
---
>        BorderTextureColor = PreferenceInt("BorderTextureColor", Default:BorderColors.count - 1,
>                                           Min:0, Max:BorderColors.count - 1)
4027c4343
<        UCIEngineDifficulty = Prefs.PreferenceInt("UCIEngineDifficulty", Default:20, Min:0, Max:20)
---
>        UCIEngineDifficulty = PreferenceInt("UCIEngineDifficulty", Default:20, Min:0, Max:20)
4036c4352
<        UCIEngineHash = Prefs.PreferenceInt("UCIEngineHash", Default:16, Min:16, Max:512)
---
>        UCIEngineHash = PreferenceInt("UCIEngineHash", Default:16, Min:16, Max:512)
4045c4361
<        UCIEngineRiskiness = Prefs.PreferenceInt("UCIEngineRiskiness", Default:0, Min:-100, Max:100)
---
>        UCIEngineRiskiness = PreferenceInt("UCIEngineRiskiness", Default:0, Min:-100, Max:100)
4054c4370
<        SearchTime = Prefs.PreferenceInt("SearchTime", Default:5, Min:5, Max:120)
---
>        SearchTime = PreferenceInt("SearchTime", Default:5, Min:5, Max:120)
4063c4379
<        MovieFPSFromSlider = Prefs.PreferenceInt("MovieFPSFromSlider", Default:10, Min:0, Max:100)
---
>        MovieFPSFromSlider = PreferenceInt("MovieFPSFromSlider", Default:10, Min:0, Max:100)
4072c4388
<        SoundOff = Prefs.PreferenceBool("SoundOff", Default:false)
---
>        SoundOff = PreferenceBool("SoundOff", Default:false)
4075c4391
<        GameSoundsButton.state = NSControl.StateValue(rawValue:BoolTo().Num(!SoundOff))
---
>        GameSoundsButton.state = NSControl.StateValue(rawValue:BoolToInt(!SoundOff))
4078c4394
<        MoveBySelectingSquares = Prefs.PreferenceBool("MoveBySelectingSquares", Default:false)
---
>        MoveBySelectingSquares = PreferenceBool("MoveBySelectingSquares", Default:false)
4081,4082c4397,4398
<        MoveBySelectingSquaresButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                                    .Num(MoveBySelectingSquares))
---
>        MoveBySelectingSquaresButton.state = NSControl.StateValue(rawValue:
>                                                                   BoolToInt(MoveBySelectingSquares))
4085c4401
<        ShowStockfishAnalysis = Prefs.PreferenceBool("ShowStockfishAnalysis", Default:false)
---
```

```
>       ShowStockfishAnalysis = PreferenceBool("ShowStockfishAnalysis", Default:false)
4088,4089c4404,4405
<       ShowStockfishAnalysisButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                             .Num(ShowStockfishAnalysis))
---
>       ShowStockfishAnalysisButton.state = NSControl.StateValue(rawValue:
>                                                             BoolToInt(ShowStockfishAnalysis))
4092c4408
<       AutoloadStockfish = Prefs.PreferenceBool("AutoloadStockfish", Default:false)
---
>       AutoloadStockfish = PreferenceBool("AutoloadStockfish", Default:false)
4098c4414
<          LoadEngine(Bundle.main.path(forResource:"stockfish-11-64", ofType:"")!)
---
>          LoadEngine(Bundle.main.path(forResource:"stockfish-10-64", ofType:"")!)
4102c4418
<       HideInfoPanel = Prefs.PreferenceBool("HideInfoPanel", Default:false)
---
>       HideInfoPanel = PreferenceBool("HideInfoPanel", Default:false)
4105c4421
<       SmallInfoPanel = Prefs.PreferenceBool("SmallInfoPanel", Default:false)
---
>       SmallInfoPanel = PreferenceBool("SmallInfoPanel", Default:false)
4108c4424
<       LargeInfoPanel = Prefs.PreferenceBool("LargeInfoPanel", Default:true)
---
>       LargeInfoPanel = PreferenceBool("LargeInfoPanel", Default:true)
4111c4427
<       HideGameClockInPanel = Prefs.PreferenceBool("HideGameClockInPanel", Default:false)
---
>       HideGameClockInPanel = PreferenceBool("HideGameClockInPanel", Default:false)
4114c4430,4431
<       DisplayClockButton.state = NSControl.StateValue(rawValue:BoolTo().Num(!HideGameClockInPanel))
---
>       DisplayClockButton.state = NSControl.StateValue(rawValue:
>                                                    BoolToInt(!HideGameClockInPanel))
4120c4437
<       HideIllustrations = Prefs.PreferenceBool("HideIllustrations", Default:false)
---
>       HideIllustrations = PreferenceBool("HideIllustrations", Default:false)
4123,4124c4440,4441
<       DisplayIllustrationsButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                            .Num(!HideIllustrations))
---
>       DisplayIllustrationsButton.state = NSControl.StateValue(rawValue:
>                                                            BoolToInt(!HideIllustrations))
4127c4444
<       HideCapturedPiecesInBoard = Prefs.PreferenceBool("HideCapturedPiecesInBoard", Default:false)
---
>       HideCapturedPiecesInBoard = PreferenceBool("HideCapturedPiecesInBoard", Default:false)
4130,4131c4447,4448
<       DisplayCapturedPiecesButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                             .Num(!HideCapturedPiecesInBoard))
---
>       DisplayCapturedPiecesButton.state = NSControl.StateValue(rawValue:
>                                                   BoolToInt(!HideCapturedPiecesInBoard))
4134c4451
<       RotateBoard = Prefs.PreferenceBool("RotateBoard", Default:false)
---
>       RotateBoard = PreferenceBool("RotateBoard", Default:false)
4137c4454
<       RotateBoardButton.state = NSControl.StateValue(rawValue:BoolTo().Num(RotateBoard))
---
>       RotateBoardButton.state = NSControl.StateValue(rawValue:BoolToInt(RotateBoard))
4140c4457
<       UseEcoBook = Prefs.PreferenceBool("UseEcoBook", Default:false)
---
>       UseEcoBook = PreferenceBool("UseEcoBook", Default:false)
4143c4460
<       EcoBookButton.state = NSControl.StateValue(rawValue:BoolTo().Num(UseEcoBook))
---
>       EcoBookButton.state = NSControl.StateValue(rawValue:BoolToInt(UseEcoBook))
4146c4463
<       AutosaveGame = Prefs.PreferenceBool("AutosaveGame", Default:false)
---
>       AutosaveGame = PreferenceBool("AutosaveGame", Default:false)
4149c4466
<       AutosaveGameButton.state = NSControl.StateValue(rawValue:BoolTo().Num(AutosaveGame))
---
>       AutosaveGameButton.state = NSControl.StateValue(rawValue:BoolToInt(AutosaveGame))
4152c4469
<       HighlightLastMoveMade = Prefs.PreferenceBool("HighlightLastMoveMade", Default:false)
---
>       HighlightLastMoveMade = PreferenceBool("HighlightLastMoveMade", Default:false)
4155,4156c4472,4473
<       HighlightLastMoveMadeButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                             .Num(HighlightLastMoveMade))
---
>       HighlightLastMoveMadeButton.state = NSControl.StateValue(rawValue:
>                                                             BoolToInt(HighlightLastMoveMade))
4159c4476
<       HighlightCapturedPieces = Prefs.PreferenceBool("HighlightCapturedPieces", Default:false)
---
>       HighlightCapturedPieces = PreferenceBool("HighlightCapturedPieces", Default:false)
4162,4163c4479,4480
<       HighlightCapturedPiecesButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                               .Num(HighlightCapturedPieces))
---
>       HighlightCapturedPiecesButton.state = NSControl.StateValue(rawValue:
>                                                               BoolToInt(HighlightCapturedPieces))
4166c4483
<       HighlightThreatenedPieces = Prefs.PreferenceBool("HighlightThreatenedPieces", Default:false)
---
>       HighlightThreatenedPieces = PreferenceBool("HighlightThreatenedPieces", Default:false)
4169,4170c4486,4487
<       HighlightThreatenedPiecesButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                                .Num(HighlightThreatenedPieces))
---
>       HighlightThreatenedPiecesButton.state = NSControl.StateValue(rawValue:
>                                                       BoolToInt(HighlightThreatenedPieces))
4173c4490
<       MultithreadUCIEngine = Prefs.PreferenceBool("MultithreadUCIEngine", Default:false)
---
>       MultithreadUCIEngine = PreferenceBool("MultithreadUCIEngine", Default:false)
4176,4177c4493,4494
<       MultithreadUCIEngineButton.state = NSControl.StateValue(rawValue:BoolTo()
<                                                            .Num(MultithreadUCIEngine))
---
>       MultithreadUCIEngineButton.state = NSControl.StateValue(rawValue:
>                                                            BoolToInt(MultithreadUCIEngine))
4183c4500
<       SavedPositionSlot = Prefs.PreferenceInt("SavedPositionSlot", Default:1, Min:1, Max:5)
---
>       SavedPositionSlot = PreferenceInt("SavedPositionSlot", Default:1, Min:1, Max:5)
4218c4535,4590
<       PreferencesSelection = Prefs.PreferenceInt("PreferencesSelection", Default:20, Min:20, Max:28)
---
```

```
>       PreferencesSelection = PreferenceInt("PreferencesSelection", Default:20, Min:20, Max:28)
>
>       // if the preferences selection is 20 //
>       if PreferencesSelection == 20
>       {
>          // set the background preferences //
>          BackgroundPreferencesAction(self)
>       }
>       // if the preferences selection is 21 //
>       else if PreferencesSelection == 21
>       {
>          // set the board preferences //
>          BoardPreferencesAction(self)
>       }
>       // if the preferences selection is 22 //
>       else if PreferencesSelection == 22
>       {
>          // set the font preferences //
>          FontPreferencesAction(self)
>       }
>       // if the preferences selection is 23 //
>       else if PreferencesSelection == 23
>       {
>          // set the stockfish preferences //
>          StockfishPreferencesAction(self)
>       }
>       // if the preferences selection is 24 //
>       else if PreferencesSelection == 24
>       {
>          // set the speed preferences //
>          SpeedPreferencesAction(self)
>       }
>       // if the preferences selection is 25 //
>       else if PreferencesSelection == 25
>       {
>          // set the clock preferences //
>          ClockPreferencesAction(self)
>       }
>       // if the preferences selection is 26 //
>       else if PreferencesSelection == 26
>       {
>          // set the movie preferences setting //
>          MoviePreferencesAction(self)
>       }
>       // if the preferences selection is 27 //
>       else if PreferencesSelection == 27
>       {
>          // set the gallery preferences setting //
>          GalleryPreferencesAction(self)
>       }
>       // if the preferences selection is 28 //
>       else if PreferencesSelection == 28
>       {
>          // set the diagram preserences setting //
>          DiagramPreferencesAction(self)
>       }
4223,4225d4594
<
<       // update to the current preferences selection //
<       PreferencesAction(self)
4247c4616
<       UpdateOpponent(Prefs.PreferenceString("Opponent", Default:"GreenChess"))
---
>       UpdateOpponent(PreferenceString("Opponent", Default:"GreenChess"))
4869c5238
<       AppNameView.stringValue = "GreenChess 8.2.8"
---
>       AppNameView.stringValue = "GreenChess 8.2.7"
4873c5242,5257
<
---
>
```

```
>
>       // remove all the board labels font items from the board labels font button //
>       BoardLabelsFontButton.removeAllItems()
>
>       // remove all the move list font items from the move list font button //
>       MoveListFontButton.removeAllItems()
>
>       // remove all the game info font items from the game info font button //
>       GameInfoFontButton.removeAllItems()
>
>       // remove all the game list font items from the game list font button //
>       GameListFontButton.removeAllItems()
>
>       // remove all the board texture items from the board textures button //
>       BoardTexturesButton.removeAllItems()
>
4875a5260,5262
>
>       // remove all the piece items from the piece set button //
>       PieceSetButton.removeAllItems()
4893,4894c5280,5292
<       // populate the piece images button titles //
<       PieceImages.PopulateButtonTitles(PieceSetButton)
---
>       // loop over all the piece sets //
>       for Index in 0 ..< PieceImages.GetNumberOfPieceSets()
>       {
>          // add the piece set item to the piece set button //
>          PieceSetButton.addItem(withTitle:PieceImages.GetPieceSetName(Index))
>       }
>
>       // loop over all the chess board names //
>       for Index in 0 ..< BoardImages.GetNumberOfBoards()
>       {
>          // add the board texture item to the board textures button //
>          BoardTexturesButton.addItem(withTitle:BoardImages.GetBoardName(Index))
>       }
4896,4897c5294,5308
<       // populate the board images button titles //
<       BoardImages.PopulateButtonTitles(BoardTexturesButton)
---
>       // loop over all the fonts //
>       for Index in 0 ..< Fonts.count
>       {
>          // add the item to the board labels font button //
>          BoardLabelsFontButton.addItem(withTitle:Fonts[Index])
>
>          // add the item to the move list font button //
>          MoveListFontButton.addItem(withTitle:Fonts[Index])
>
>          // add the item to the game info font button //
>          GameInfoFontButton.addItem(withTitle:Fonts[Index])
>
>          // add the item to the game list font button //
>          GameListFontButton.addItem(withTitle:Fonts[Index])
>       }
4958,4960c5369,5395
<       // initialize the preferences layouts //
<       InitializePreferenceLayouts()
<
---
>       // 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 gallery preferences layout //
>       GalleryPreferencesLayout()
>
>       // update the diagram preferences layout //
>       DiagramPreferencesLayout()
>
>       // update the board preferences layout //
>       BoardPreferencesLayout()
>
>       // update the background preferences layout //
>       BackgroundPreferencesLayout()
>
>       // update the font preferences layout //
>       FontPreferencesLayout()
>
4983c5418
<       PGNMovesTableView.backgroundColor = NSColor.white.withAlphaComponent(0.25)
---
>       PGNMovesTableView.backgroundColor = NSColor.white.withAlphaComponent(0.2)
4986,5001c5421
<       GameInfoTableView.backgroundColor = NSColor.white.withAlphaComponent(0.25)
<
<       // initialize the game gallery tabs //
<       GalleryTabs.Initialize([GameGalleryLeft, GameGalleryRight, GameGalleryMovie,
<                               GameGallerySnapshot, GameGalleryErase, GameGalleryBlueColor,
<                               GameGalleryCyanColor, GameGalleryMagentaColor, GameGalleryGreenColor,
<                               GameGalleryRedColor, GameGalleryYellowColor, GameGalleryBlackColor,
<                               GameGalleryWhiteColor, GameGalleryUndoPath, GameGallerySlider],
<                               ThePlayButton:GameGalleryReviewButton)
<
<       // initialize the game explorer tabs //
<       GameExplorerTabs.Initialize([EnterAnalysisModeButton, EnterPositionEditorModeButton,
<                                    EnterDiagramModeButton, GameExplorerSnapshotButton,
<                                    GameExplorerMovieButton, ExitGameExplorerButton,
<                                    GameExplorerSlider, GameExplorerLeft, GameExplorerRight],
<                                    ThePlayButton:GameReviewButton)
---
>       GameInfoTableView.backgroundColor = NSColor.white.withAlphaComponent(0.2)
5092a5513,5515
>       // set the board labels font size from the board labels font sizes value //
>       BoardLabelsSize = CGFloat(16 + 4 * BoardLabelsSizes)
>
5094,5095c5517,5525
<       RankLabelOffset = Dims.GetBorderWidth() + (Dims.GetSquareSize(2) - BoardLabelsFont
<                         .GetFontSize()) * 0.5
---
>       RankLabelOffset = Dims.GetBorderWidth() + (Dims.GetSquareSize(2) - BoardLabelsSize) * 0.5
>
>       // get the board label font //
>       BoardLabelsFont = GetFont(BoardLabelsFontType, Size:Scale * BoardLabelsSize, Bold:
>                                 BoardLabelsBold)
>
>       // set the board label attributes //
>       BoardLabelsAttributes = [NSAttributedString.Key.font:BoardLabelsFont, NSAttributedString.Key
>                                .foregroundColor:BoardLabelsColorView.GetCurrentHue()]
5101c5531
<          Index = abs(BoolTo().Num(RotateBoard) * 7 - LabelIndex)
---
```

```
>          Index = abs(BoolToInt(RotateBoard) * 7 - LabelIndex)
5104c5534,5535
<          FileLabel = BoardLabelsFont.GetString(FileSymbols[Index], White:false)
---
>          FileLabel = NSAttributedString(string:FileSymbols[Index], attributes:BoardLabelsAttributes
>                                         as? [NSAttributedString.Key:Any])
5107c5538,5539
<          RankLabel = BoardLabelsFont.GetString(RankSymbols[Index], White:false)
---
>          RankLabel = NSAttributedString(string:RankSymbols[Index], attributes:BoardLabelsAttributes
>                                         as? [NSAttributedString.Key:Any])
5112c5544
<                        BoardLabelsFont.GetFontSize() / 1.8)))
---
>                        BoardLabelsSize / 1.8)))
5157a5590,5629
>    // get the font from the specified font type, size, and bold setting //
>    func GetFont(_ Type:Int, Size:CGFloat, Bold:Bool) -> NSFont
>    {
>       // if font is bold //
>       if Bold
>       {
>          // assign bold to the font bold string //
>          FontBoldString = " Bold"
>       }
>       // font is not bold //
>       else
>       {
>          // clear the font bold string //
>          FontBoldString = ""
>       }
>
>       // if the font exists //
>       if NSFont(name:Fonts[Type] + FontBoldString, size:Size) != nil
>       {
>          // return the font //
>          return NSFont(name:Fonts[Type] + FontBoldString, size:Size)!
>       }
>       // the font doesn't exist //
>       else
>       {
>          // if the font is bold //
>          if Bold
>          {
>             // return a default bold system font //
>             return NSFont.boldSystemFont(ofSize:Size)
>          }
>          // the font is not bold //
>          else
>          {
>             // return a default system font //
>             return NSFont.boldSystemFont(ofSize:Size)
>          }
>       }
>    }
>
6069c6541
<    @IBAction func UpdateDiagramColor(_:AnyObject)
---
>    func UpdateDiagramColor(_ Color:Int)
6074,6082c6546,6606
<       // if the color is in the colors dictionary //
<       if Colors[MainToolbar!.selectedItemIdentifier!.rawValue] != nil
<       {
<          // set the diagram cursor //
<          DiagramCursor = Markers.Cursor(Colors[MainToolbar!.selectedItemIdentifier!.rawValue]!)
<
<          // set the current diagram color //
<          DiagramColor = Markers.Color(Colors[MainToolbar!.selectedItemIdentifier!.rawValue]!)
<       }
---
>       // set the diagram cursor //
>       DiagramCursor = Markers.Cursor(Color)
>
>       // set the current diagram color //
>       DiagramColor = Markers.Color(Color)
>    }
>
>    // the diagram blue color action //
>    @IBAction func DiagramBlueColorAction(_:AnyObject)
>    {
>       // update the diagram color to blue //
>       UpdateDiagramColor(2)
>    }
>
>    // the diagram cyan color action //
>    @IBAction func DiagramCyanColorAction(_:AnyObject)
>    {
>       // update the diagram color to cyan //
>       UpdateDiagramColor(5)
>    }
>
>    // the diagram magenta color action //
>    @IBAction func DiagramMagentaColorAction(_:AnyObject)
>    {
>       // update the diagram color to magenta //
>       UpdateDiagramColor(6)
>    }
>
>    // the diagram white color action //
>    @IBAction func DiagramWhiteColorAction(_:AnyObject)
>    {
>       // update the diagram color to white //
>       UpdateDiagramColor(3)
>    }
>
>    // the diagram black color action //
>    @IBAction func DiagramBlackColorAction(_:AnyObject)
>    {
>       // update the diagram color to black //
>       UpdateDiagramColor(4)
>    }
>
>    // the diagram yellow color action //
>    @IBAction func DiagramYellowColorAction(_:AnyObject)
>    {
>       // update the diagram color to yellow //
>       UpdateDiagramColor(7)
>    }
>
>    // the diagram green color action //
>    @IBAction func DiagramGreenColorAction(_:AnyObject)
>    {
>       // update the diagram color to green //
>       UpdateDiagramColor(1)
>    }
>
```

```
>      // the diagram red color action //
>      @IBAction func DiagramRedColorAction(_:AnyObject)
>      {
>         // update the diagram color to red //
>         UpdateDiagramColor(0)
6105,6106c6629,6630
<      // update the gallery color //
<      @IBAction func UpdateGalleryColor(_:AnyObject)
---
>      // the gallery blue color action //
>      @IBAction func GalleryBlueColor(_:AnyObject)
6108,6109c6632,6633
<         // update the gallery from the gallery toolbar selected color //
<         GameGalleryView.UpdateGalleryColor(GalleryToolbar!.selectedItemIdentifier!.rawValue)
---
>         // update the gallery diagram color to blue //
>         GameGalleryView.UpdateGalleryColor(2)
6111c6635,6684
<
---
>
>      // the gallery cyan color action //
>      @IBAction func GalleryCyanColor(_:AnyObject)
>      {
>         // update the gallery diagram color to cyan //
>         GameGalleryView.UpdateGalleryColor(5)
>      }
>
>      // the gallery yellow color action //
>      @IBAction func GalleryYellowColor(_:AnyObject)
>      {
>         // update the gallery diagram color to yellow //
>         GameGalleryView.UpdateGalleryColor(7)
>      }
>
>      // the gallery black color action //
>      @IBAction func GalleryBlackColor(_:AnyObject)
>      {
>         // update the gallery diagram color to black //
>         GameGalleryView.UpdateGalleryColor(4)
>      }
>
>      // the gallery white color action //
>      @IBAction func GalleryWhiteColor(_:AnyObject)
>      {
>         // update the gallery diagram color to white //
>         GameGalleryView.UpdateGalleryColor(3)
>      }
>
>      // the gallery magenta color action //
>      @IBAction func GalleryMagentaColor(_:AnyObject)
>      {
>         // update the gallery diagram color to magenta //
>         GameGalleryView.UpdateGalleryColor(6)
>      }
>
>      // the gallery green color action //
>      @IBAction func GalleryGreenColor(_:AnyObject)
>      {
>         // update the gallery diagram color to green //
>         GameGalleryView.UpdateGalleryColor(1)
>      }
>
>      // the gallery red color action //
>      @IBAction func GalleryRedColor(_:AnyObject)
>      {
>         // update the gallery diagram color to red //
>         GameGalleryView.UpdateGalleryColor(0)
>      }
>
6423c6996
<            LoadEngine(Bundle.main.path(forResource:"stockfish-11-64", ofType:"")!)
---
>            LoadEngine(Bundle.main.path(forResource:"stockfish-10-64", ofType:"")!)
6455c7028
<         Plies = BoolTo().Num(Attributes.FENColor == "b")
---
>         Plies = BoolToInt(Attributes.FENColor == "b")
7188a7762,7764
>
>         // update the position editor toolbar buttons //
>         UpdatePositionEditorToolbarButtons()
7703,7714c8279,8296
<
<         // save the move list font preferences //
<         MoveListFont.SavePreferences()
<
<         // save the game info font preferences //
<         GameInfoFont.SavePreferences()
<
<         // save the game list font preferences //
<         GameListFont.SavePreferences()
<
<         // save the board labels font preferences //
<         BoardLabelsFont.SavePreferences()
---
>
>         // save the board labels font type to preferences //
>         AppPreferences.set(BoardLabelsFontType, forKey:"BoardLabelsFontType")
>
>         // save the board labels bold setting to preferences //
>         AppPreferences.set(BoardLabelsBold, forKey:"BoardLabelsBold")
>
>         // save the board labels font sizes setting to preferences //
>         AppPreferences.set(BoardLabelsSizes, forKey:"BoardLabelsSizes")
>
>         // save the move list font type to preferences //
>         AppPreferences.set(MoveListFontType, forKey:"MoveListFontType")
>
>         // save the move list bold setting to preferences //
>         AppPreferences.set(MoveListBold, forKey:"MoveListBold")
>
>         // save the move list font sizes setting to preferences //
>         AppPreferences.set(MoveListSizes, forKey:"MoveListSizes")
7717,7718c8299,8330
<         Prefs.Save(NumberOfGridRows, Key:"NumberOfGridRows")
<
```

```
---
>       AppPreferences.set(NumberOfGridRows, forKey:"NumberOfGridRows")
>
>       // save the game info font type to preferences //
>       AppPreferences.set(GameInfoFontType, forKey:"GameInfoFontType")
>
>       // save the game info bold setting to preferences //
>       AppPreferences.set(GameInfoBold, forKey:"GameInfoBold")
>
>       // save the game info font sizes setting to preferences //
>       AppPreferences.set(GameInfoSizes, forKey:"GameInfoSizes")
>
>       // save the game list font type to preferences //
>       AppPreferences.set(GameListFontType, forKey:"GameListFontType")
>
>       // save the game list bold setting to preferences //
>       AppPreferences.set(GameListBold, forKey:"GameListBold")
>
>       // save the game list font sizes setting to preferences //
>       AppPreferences.set(GameListSizes, forKey:"GameListSizes")
>
>       // save the chosen label hue //
>       AppPreferences.set(LabelHue, forKey:"LabelHue")
>
>       // save the chosen info hue //
>       AppPreferences.set(InfoHue, forKey:"InfoHue")
>
>       // save the chosen move list hue //
>       AppPreferences.set(MoveListHue, forKey:"MoveListHue")
>
>       // save the chosen game list hue //
>       AppPreferences.set(GameListHue, forKey:"GameListHue")
>
7720c8332
<       Prefs.Save(WindowBackgroundHue, Key:"WindowBackgroundHue")
---
>       AppPreferences.set(WindowBackgroundHue, forKey:"WindowBackgroundHue")
7723c8335
<       Prefs.Save(GameClockHue, Key:"GameClockHue")
---
>       AppPreferences.set(GameClockHue, forKey:"GameClockHue")
7726c8338
<       Prefs.Save(GameClockBackgroundHue, Key:"GameClockBackgroundHue")
---
>       AppPreferences.set(GameClockBackgroundHue, forKey:"GameClockBackgroundHue")
7729c8341
<       Prefs.Save(ClockBackgroundVisibility, Key:"ClockBackgroundVisibility")
---
>       AppPreferences.set(ClockBackgroundVisibility, forKey:"ClockBackgroundVisibility")
7732c8344
<       Prefs.Save(BoardType, Key:"BoardType")
---
>       AppPreferences.set(BoardType, forKey:"BoardType")
7735c8347
<       Prefs.Save(ChessPieceSetType, Key:"ChessPieceSetType")
---
>       AppPreferences.set(ChessPieceSetType, forKey:"ChessPieceSetType")
7738c8350
<       Prefs.Save(BorderTextureColor, Key:"BorderTextureColor")
---
>       AppPreferences.set(BorderTextureColor, forKey:"BorderTextureColor")
7741c8353
<       Prefs.Save(BorderVisibility, Key:"BorderVisibility")
---
>       AppPreferences.set(BorderVisibility, forKey:"BorderVisibility")
7744c8356
<       Prefs.Save(BoardTintVisibility, Key:"BoardTintVisibility")
---
>       AppPreferences.set(BoardTintVisibility, forKey:"BoardTintVisibility")
7747c8359
<       Prefs.Save(WindowTextureOpacity, Key:"WindowTextureOpacity")
---
>       AppPreferences.set(WindowTextureOpacity, forKey:"WindowTextureOpacity")
7750c8362
<       Prefs.Save(BoardRoughness, Key:"BoardRoughness")
---
>       AppPreferences.set(BoardRoughness, forKey:"BoardRoughness")
7753c8365
<       Prefs.Save(SquareBorder, Key:"SquareBorder")
---
>       AppPreferences.set(SquareBorder, forKey:"SquareBorder")
7756c8368
<       Prefs.Save(BoardTintHue, Key:"BoardTintHue")
---
>       AppPreferences.set(BoardTintHue, forKey:"BoardTintHue")
7759c8371
<       Prefs.Save(HighlightLastMoveMade, Key:"HighlightLastMoveMade")
---
>       AppPreferences.set(HighlightLastMoveMade, forKey:"HighlightLastMoveMade")
7762c8374
<       Prefs.Save(HighlightThreatenedPieces, Key:"HighlightThreatenedPieces")
---
>       AppPreferences.set(HighlightThreatenedPieces, forKey:"HighlightThreatenedPieces")
7765c8377
<       Prefs.Save(HighlightCapturedPieces, Key:"HighlightCapturedPieces")
---
>       AppPreferences.set(HighlightCapturedPieces, forKey:"HighlightCapturedPieces")
7768c8380
<       Prefs.Save(MultithreadUCIEngine, Key:"MultithreadUCIEngine")
---
>       AppPreferences.set(MultithreadUCIEngine, forKey:"MultithreadUCIEngine")
7771c8383
<       Prefs.Save(AutosaveGame, Key:"AutosaveGame")
---
>       AppPreferences.set(AutosaveGame, forKey:"AutosaveGame")
7774c8386
<       Prefs.Save(ShowStockfishAnalysis, Key:"ShowStockfishAnalysis")
---
>       AppPreferences.set(ShowStockfishAnalysis, forKey:"ShowStockfishAnalysis")
7777c8389
<       Prefs.Save(AnimationRate, Key:"AnimationRate")
---
>       AppPreferences.set(AnimationRate, forKey:"AnimationRate")
7780c8392
<       Prefs.Save(GalleryAnimationRate, Key:"GalleryAnimationRate")
---
>       AppPreferences.set(GalleryAnimationRate, forKey:"GalleryAnimationRate")
7783c8395
<       Prefs.Save(HideCapturedPiecesInBoard, Key:"HideCapturedPiecesInBoard")
---
>       AppPreferences.set(HideCapturedPiecesInBoard, forKey:"HideCapturedPiecesInBoard")
7786c8398
<       Prefs.Save(RotateBoard, Key:"RotateBoard")
---
>       AppPreferences.set(RotateBoard, forKey:"RotateBoard")
7789c8401
<       Prefs.Save(HideGameClockInPanel, Key:"HideGameClockInPanel")
---
>       AppPreferences.set(HideGameClockInPanel, forKey:"HideGameClockInPanel")
7792c8404
<       Prefs.Save(IllustrationSize, Key:"IllustrationSize")
---
```

```
>        AppPreferences.set(IllustrationSize, forKey:"IllustrationSize")
7795c8407
<        Prefs.Save(HideIllustrations, Key:"HideIllustrations")
---
>        AppPreferences.set(HideIllustrations, forKey:"HideIllustrations")
7798c8410
<        Prefs.Save(UCIEngineDifficulty, Key:"UCIEngineDifficulty")
---
>        AppPreferences.set(UCIEngineDifficulty, forKey:"UCIEngineDifficulty")
7801c8413
<        Prefs.Save(UCIEngineHash, Key:"UCIEngineHash")
---
>        AppPreferences.set(UCIEngineHash, forKey:"UCIEngineHash")
7804c8416
<        Prefs.Save(UCIEngineRiskiness, Key:"UCIEngineRiskiness")
---
>        AppPreferences.set(UCIEngineRiskiness, forKey:"UCIEngineRiskiness")
7807c8419
<        Prefs.Save(SearchTime, Key:"SearchTime")
---
>        AppPreferences.set(SearchTime, forKey:"SearchTime")
7810c8422,8425
<        Prefs.Save(UseEcoBook, Key:"UseEcoBook")
---
>        AppPreferences.set(UseEcoBook, forKey:"UseEcoBook")
>
>        // get the current preferences selection //
>        PreferencesSelection = Int(PreferencesToolbar.selectedItemIdentifier!.rawValue)!
7813c8428
<        Prefs.Save(PreferencesSelection, Key:"PreferencesSelection")
---
>        AppPreferences.set(PreferencesSelection, forKey:"PreferencesSelection")
7816c8431
<        Prefs.Save(SoundOff, Key:"SoundOff")
---
>        AppPreferences.set(SoundOff, forKey:"SoundOff")
7819c8434
<        Prefs.Save(MoveBySelectingSquares, Key:"MoveBySelectingSquares")
---
>        AppPreferences.set(MoveBySelectingSquares, forKey:"MoveBySelectingSquares")
7822c8437
<        Prefs.Save(MovieFPSFromSlider, Key:"MovieFPSFromSlider")
---
>        AppPreferences.set(MovieFPSFromSlider, forKey:"MovieFPSFromSlider")
8064a8680,8729
>
>     // set the board labels font //
>     @IBAction func SetBoardLabelsFont(_:AnyObject)
>     {
>        // get the board labels font type from the board labels font button //
>        BoardLabelsFontType = BoardLabelsFontButton.indexOfSelectedItem
>
>        // refresh the board and pieces //
>        RefreshBoardAndPieces(2, UpdateBoard:true, Display:true)
>     }
>
>     // set the move list font //
>     @IBAction func SetMoveListFont(_:AnyObject)
>     {
>        // get the move list font type from the move list font button //
>        MoveListFontType = MoveListFontButton.indexOfSelectedItem
>
>        // update the move list //
>        UpdateMoveList()
>     }
>
>     // set the game info font //
>     @IBAction func SetGameInfoFont(_:AnyObject)
>     {
>        // get the game info font type from the game info font button //
>        GameInfoFontType = GameInfoFontButton.indexOfSelectedItem
>
>        // update the game info list //
>        UpdateGameInfoList()
>     }
>
>     // update the chess piece set type //
>     @IBAction func UpdatePieceSetType(_:AnyObject)
>     {
>        // 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
>
>           // update the piece set type //
>           PieceImages.UpdatePieceSet(ChessPieceSetType)
>
>           // refresh the board and pieces //
>           RefreshBoardAndPieces(2, UpdateBoard:false, Display:true)
>
>           // update the position editor toolbar buttons //
>           UpdatePositionEditorToolbarButtons()
>        }
>     }
8066,8067c8731,8732
<     // update the game list font //
<     @IBAction func UpdateGameListFont(_:AnyObject)
---
>     // update the board type //
>     @IBAction func UpdateBoardTypeLayer(_:AnyObject)
8069,8070c8734,8742
<        // update the game list font //
<        GameListFont.UpdateFont()
---
>        // if the board type is different //
>        if BoardTexturesButton.indexOfSelectedItem != BoardType
>        {
>           // get the board type from the board textures button //
>           BoardType = BoardTexturesButton.indexOfSelectedItem
>
>           // refresh the board and pieces //
>           RefreshBoardAndPieces(2, UpdateBoard:true, Display:true)
>        }
8073,8074c8745,8746
<     // update the move list font //
<     @IBAction func UpdateMoveListFont(_:AnyObject)
---
>     // set the game list font //
>     @IBAction func SetGameListFont(_:AnyObject)
8076,8077c8748,8783
<        // update the move list font //
<        MoveListFont.UpdateFont()
---
```

```
>       // get the game list font type from the game list button //
>       GameListFontType = GameListFontButton.indexOfSelectedItem
>    }
>
>    // set the move list bold type //
>    @IBAction func SetMoveListBold(_:AnyObject)
>    {
>       // get the move list bold type from the move list bold button //
>       MoveListBold = IntToBool(MoveListBoldButton.indexOfSelectedItem)
>
>       // update the move list //
>       UpdateMoveList()
>    }
>
>    // set the game info bold type //
>    @IBAction func SetGameInfoBold(_:AnyObject)
>    {
>       // get the game info bold type from the game info bold button //
>       GameInfoBold = IntToBool(GameInfoBoldButton.indexOfSelectedItem)
>
>       // update the game info list //
>       UpdateGameInfoList()
>    }
>
>    // set the game list bold type //
>    @IBAction func SetGameListBold(_:AnyObject)
>    {
>       // get the game list bold type from the game list bold button //
>       GameListBold = IntToBool(GameListBoldButton.indexOfSelectedItem)
>    }
>
>    // set the board labels bold setting //
>    @IBAction func SetBoardLabelsBold(_:AnyObject)
>    {
>       // get the board labels bold setting from the board labels bold button //
>       BoardLabelsBold = IntToBool(BoardLabelsBoldButton.indexOfSelectedItem)
8078a8785,8794
>       // refresh the board and pieces //
>       RefreshBoardAndPieces(2, UpdateBoard:true, Display:true)
>    }
>
>    // set the move list font sizes setting //
>    @IBAction func SetMoveListSizes(_:AnyObject)
>    {
>       // get the move list font sizes setting from the move list font sizes button //
>       MoveListSizes = MoveListSizesButton.indexOfSelectedItem
>
8083,8084c8799,8800
<    // update the board labels font //
<    @IBAction func UpdateBoardLabelsFont(_:AnyObject)
---
>    // set the game info sizes setting //
>    @IBAction func SetGameInfoSizes(_:AnyObject)
8086,8087c8802,8820
<       // update the board labels font //
<       BoardLabelsFont.UpdateFont()
---
>       // get the game info sizes setting from the game info fonts sizes button //
>       GameInfoSizes = GameInfoSizesButton.indexOfSelectedItem
>
>       // update the game info list //
>       UpdateGameInfoList()
>    }
>
>    // set the game list sizes setting //
>    @IBAction func SetGameListSizes(_:AnyObject)
>    {
>       // get the game list sizes setting from the game info fonts sizes button //
>       GameListSizes = GameListSizesButton.indexOfSelectedItem
>    }
>
>    // set the board labels font sizes setting //
>    @IBAction func SetBoardLabelsSizes(_:AnyObject)
>    {
>       // get board labels font sizes setting from the board labels font sizes button //
>       BoardLabelsSizes = BoardLabelsSizesButton.indexOfSelectedItem
8093,8094c8826,9424
<    // update the game info font //
<    @IBAction func UpdateGameInfoFont(_:AnyObject)
---
>    // set the board labels font color //
>    func SetBoardLabelsFontColor()
>    {
>       // get the label color from the current color //
>       LabelHue = BoardLabelsColorView.GetHueValue()
>    }
>
>    // set the game info font color //
>    func SetInfoFontColor()
>    {
>       // get the game info color from the current color //
>       InfoHue = GameInfoColorView.GetHueValue()
>
>       // update the game info text //
>       UpdateGameInfoList()
>    }
>
>    // set the move list font color //
>    func SetMoveListFontColor()
>    {
>       // get the move list color from the current color //
>       MoveListHue = MoveListColorView.GetHueValue()
>
>       // update the move list //
>       UpdateMoveList()
>    }
>
>    // set the game clock background color //
>    func SetGameClockBackgroundColor()
>    {
>       // set the game clock background color from the clock background color view //
>       GameClockBackgroundHue = ClockBackgroundColorView.GetHueValue()
>
>       // update the white time clock background color //
>       WhiteTimeClock.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
>
>       // update the black time clock background color //
>       BlackTimeClock.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
>
>       // update the time clock preview background color //
>       TimeClockPreview.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
>    }
>
```

```
>     // set the game clock color //
>     func SetGameClockColor()
>     {
>        // set the game clock color from the clock color view //
>        GameClockHue = ClockColorView.GetHueValue()
>
>        // update the white time clock border color //
>        WhiteTimeClock.UpdateClockBorderColor(ClockColorView.GetCurrentHue())
>
>        // update the black time clock border color //
>        BlackTimeClock.UpdateClockBorderColor(ClockColorView.GetCurrentHue())
>
>        // update the time clock border preview color //
>        TimeClockPreview.UpdateClockBorderColor(ClockColorView.GetCurrentHue())
>     }
>
>     // set the game list font color //
>     func SetGameListFontColor()
>     {
>        // get game list color from the current color //
>        GameListHue = GameListColorView.GetHueValue()
>     }
>
>     // update the board preferences to the given state //
>     func BoardPreferencesToState(_ State:Bool)
>     {
>        // update the board textures button state //
>        BoardTexturesButton.isHidden = State
>
>        // update the piece set button state //
>        PieceSetButton.isHidden = State
>
>        // update the border visibility slider state //
>        BorderVisibilitySlider.isHidden = State
>
>        // update the board tint visibility slider state //
>        BoardTintVisibilitySlider.isHidden = State
>
>        // update the board roughness slider state //
>        BoardRoughnessSlider.isHidden = State
>
>        // update the square border opacity slider state //
>        SquareBorderOpacitySlider.isHidden = State
>
>        // update the board tint color slider state //
>        BoardTintColorSlider.isHidden = State
>
>        // update the border visibility view state //
>        BorderVisibilityView.isHidden = State
>
>        // update the board tint visibility view state //
>        BoardTintVisibilityView.isHidden = State
>
>        // update the board roughness view state //
>        BoardRoughnessView.isHidden = State
>
>        // update the square border view state //
>        SquareBorderView.isHidden = State
>
>        // update the board tint color view state //
>        BoardTintColorView.isHidden = State
>
>        // update the border texture slider state //
>        BorderTextureSlider.isHidden = State
>
>        // update the border texture view state //
>        BorderTextureView.isHidden = State
>
>        // update the board preview image state //
>        BoardPreview.isHidden = State
>
>        // udate the autosave game button state //
>        AutosaveGameButton.isHidden = State
>
>        // update the game sound button state //
>        GameSoundsButton.isHidden = State
>
>        // update the move by selecting squares button state //
>        MoveBySelectingSquaresButton.isHidden = State
>
>        // update the display captured pieces button state //
>        DisplayCapturedPiecesButton.isHidden = State
>
>        // update the rotate board button state //
>        RotateBoardButton.isHidden = State
>
>        // update the opening book button state //
>        EcoBookButton.isHidden = State
>
>        // update the highlight last move made button state //
>        HighlightLastMoveMadeButton.isHidden = State
>
>        // update the highlight captured pieces button state //
>        HighlightCapturedPiecesButton.isHidden = State
>
>        // update the highlight threatened pieces button state //
>        HighlightThreatenedPiecesButton.isHidden = State
>     }
>
>     // set the background preferences to the given state //
>     func BackgroundPreferencesToState(_ State:Bool)
>     {
>        // update the window texture opacity slider state //
>        WindowTextureOpacitySlider.isHidden = State
>
>        // update the window texture opacity view state //
>        WindowTextureOpacityView.isHidden = State
>
>        // update the window color preview state //
>        WindowColorPreview.isHidden = State
>
>        // update the window colors background text field state //
>        WindowColorsBackgroundField.isHidden = State
>
>        // update the window background color view state //
>        WindowBackgroundColorView.isHidden = State
>     }
>
>     // update the font preference states //
>     func FontPreferencesToState(_ State:Bool)
>     {
>        // update the board labels font button state //
>        BoardLabelsFontButton.isHidden = State
```

```
>
>         // update the board labels bold button state //
>         BoardLabelsBoldButton.isHidden = State
>
>         // update the board labels font sizes button state //
>         BoardLabelsSizesButton.isHidden = State
>
>         // update the move list font button state //
>         MoveListFontButton.isHidden = State
>
>         // update the grid rows button state //
>         GridRowsButton.isHidden = State
>
>         // update the move list bold button state //
>         MoveListBoldButton.isHidden = State
>
>         // update the move list sizes button state //
>         MoveListSizesButton.isHidden = State
>
>         // update the game info font button state //
>         GameInfoFontButton.isHidden = State
>
>         // update the game info bold button state //
>         GameInfoBoldButton.isHidden = State
>
>         // update the game info sizes button state //
>         GameInfoSizesButton.isHidden = State
>
>         // update the game list font button state //
>         GameListFontButton.isHidden = State
>
>         // update the game list bold button state //
>         GameListBoldButton.isHidden = State
>
>         // update the game list sizes button state //
>         GameListSizesButton.isHidden = State
>
>         // update the board labels field state //
>         BoardLabelsField.isHidden = State
>
>         // update the move list field state //
>         MoveListField.isHidden = State
>
>         // update the game info list field state //
>         GameInfoField.isHidden = State
>
>         // update the game list field state //
>         GameListField.isHidden = State
>
>         // update the board labels color view state //
>         BoardLabelsColorView.isHidden = State
>
>         // update the move list color view state //
>         MoveListColorView.isHidden = State
>
>         // update the game info color view state //
>         GameInfoColorView.isHidden = State
>
>         // update the game list color view state //
>         GameListColorView.isHidden = State
>     }
>
>     // update the stockfish preference states //
>     func StockfishPreferencesToState(_ State:Bool)
>     {
>         // update the show the stockfish analysis button state //
>         ShowStockfishAnalysisButton.isHidden = State
>
>         // update the multithread stockfish button state //
>         MultithreadUCIEngineButton.isHidden = State
>
>         // update the uci engine difficulty slider state //
>         UCIEngineDifficultySlider.isHidden = State
>
>         // update the uci engine difficulty view state //
>         UCIEngineDifficultyView.isHidden = State
>
>         // update the uci engine hash slider state //
>         UCIEngineHashSlider.isHidden = State
>
>         // update the uci engine hash view state //
>         UCIEngineHashView.isHidden = State
>
```

```
>       // update the uci engine riskiness slider state //
>       UCIEngineRiskinessSlider.isHidden = State
>
>       // update the uci engine riskiness view state //
>       UCIEngineRiskinessView.isHidden = State
>    }
>
>    // set the board preferences layout //
>    func BoardPreferencesLayout()
>    {
>       // update the board preview frame //
>       BoardPreview.frame = CGRect(x:30, y:200, width:400, height:400)
>
>       // update the board textures button frame //
>       BoardTexturesButton.frame = CGRect(x:444, y:579, width:165, height:21)
>
>       // update the piece set button frame //
>       PieceSetButton.frame = CGRect(x:444, y:540, width:165, height:21)
>
>       // update the autosave game button frame //
>       AutosaveGameButton.frame = CGRect(x:445, y:501, width:200, height:20)
>
>       // update the game sounds button frame //
>       GameSoundsButton.frame = CGRect(x:445, y:463, width:200, height:20)
>
>       // update the opening book button frame //
>       EcoBookButton.frame = CGRect(x:445, y:425, width:200, height:20)
>
>       // update the highlight last move made button frame //
>       HighlightLastMoveMadeButton.frame = CGRect(x:445, y:387, width:200, height:20)
>
>       // update the highlight captured pieces button frame //
>       HighlightCapturedPiecesButton.frame = CGRect(x:445, y:349, width:200, height:20)
>
>       // update the highlight threatened pieces button frame //
>       HighlightThreatenedPiecesButton.frame = CGRect(x:445, y:311, width:200, height:20)
>
>       // update the move by selecting squares button frame //
>       MoveBySelectingSquaresButton.frame = CGRect(x:445, y:273, width:200, height:20)
>
>       // update the display captured pieces button frame //
>       DisplayCapturedPiecesButton.frame = CGRect(x:445, y:235, width:200, height:20)
>
>       // update the rotate board button frame //
>       RotateBoardButton.frame = CGRect(x:445, y:197, width:200, height:20)
>
>       // update the border texture view frame //
>       BorderTextureView.frame = CGRect(x:445, y:149, width:200, height:20)
>
>       // update the border texture slider frame //
>       BorderTextureSlider.frame = CGRect(x:30, y:152, width:400, height:13)
>
>       // update the border visibility view frame //
>       BorderVisibilityView.frame = CGRect(x:445, y:123, width:200, height:20)
>
>       // update the border visibility slider frame //
>       BorderVisibilitySlider.frame = CGRect(x:30, y:126, width:400, height:13)
>
>       // update the square border view frame //
>       SquareBorderView.frame = CGRect(x:445, y:97, width:200, height:20)
>
>       // update the square border opacity slider frame //
>       SquareBorderOpacitySlider.frame = CGRect(x:30, y:100, width:400, height:13)
>
>       // update the board roughness view frame //
>       BoardRoughnessView.frame = CGRect(x:445, y:71, width:200, height:20)
>
>       // update the board roughness slider frame //
>       BoardRoughnessSlider.frame = CGRect(x:30, y:74, width:400, height:13)
>
>       // update the board tint color view frame //
>       BoardTintColorView.frame = CGRect(x:445, y:45, width:200, height:20)
>
>       // update the board tint color slider frame //
>       BoardTintColorSlider.frame = CGRect(x:30, y:48, width:400, height:13)
>
>       // update the board tint visibility view frame //
>       BoardTintVisibilityView.frame = CGRect(x:445, y:19, width:200, height:20)
>
>       // update the board tint visibility slider frame //
>       BoardTintVisibilitySlider.frame = CGRect(x:30, y:22, width:400, height:13)
>    }
>
>    // update the background preferences layout //
>    func BackgroundPreferencesLayout()
>    {
>       // update the window texture opacity slider frame //
>       WindowTextureOpacitySlider.frame = CGRect(x:30, y:585, width:600, height:13)
>
>       // update the window texture opacity view frame //
>       WindowTextureOpacityView.frame = CGRect(x:30, y:567, width:600, height:20)
>
>       // update the window background color text field frame //
>       WindowColorsBackgroundField.frame = CGRect(x:230, y:540, width:200, height:20)
>
>       // update the window background color view frame //
>       WindowBackgroundColorView.frame = CGRect(x:30, y:500, width:600, height:40)
>
>       // update the window color preview frame //
>       WindowColorPreview.frame = CGRect(x:230, y:270, width:200, height:200)
>    }
>
>    // update the font preferences layout //
>    func FontPreferencesLayout()
>    {
>       // update the board labels field frame //
>       BoardLabelsField.frame = CGRect(x:230, y:585, width:200, height:20)
>
>       // update the board labels font button frame //
>       BoardLabelsFontButton.frame = CGRect(x:230, y:560, width:200, height:21)
>
>       // update the board labels bold button frame //
>       BoardLabelsBoldButton.frame = CGRect(x:230, y:535, width:200, height:21)
>
>       // update the board labels font sizes button frame //
>       BoardLabelsSizesButton.frame = CGRect(x:230, y:510, width:200, height:21)
>
>       // update the board labels color view frame //
>       BoardLabelsColorView.frame = CGRect(x:30, y:470, width:600, height:40)
>
>       // update the game info field frame //
>       GameInfoField.frame = CGRect(x:230, y:435, width:200, height:20)
>
>       // update the game info font button frame //
>       GameInfoFontButton.frame = CGRect(x:230, y:410, width:200, height:21)
>
```

```
>        // update the game info bold button frame //
>        GameInfoBoldButton.frame = CGRect(x:230, y:385, width:200, height:21)
>
>        // update the game info sizes button frame //
>        GameInfoSizesButton.frame = CGRect(x:230, y:360, width:200, height:21)
>
>        // update the game info color view frame //
>        GameInfoColorView.frame = CGRect(x:30, y:320, width:600, height:40)
>
>        // update the move list field frame //
>        MoveListField.frame = CGRect(x:230, y:285, width:200, height:20)
>
>        // update the move list font button frame //
>        MoveListFontButton.frame = CGRect(x:230, y:260, width:200, height:21)
>
>        // update the move list bold button frame //
>        MoveListBoldButton.frame = CGRect(x:230, y:235, width:200, height:21)
>
>        // update the move list font sizes button frame //
>        MoveListSizesButton.frame = CGRect(x:230, y:210, width:200, height:21)
>
>        // update the move list color view frame //
>        MoveListColorView.frame = CGRect(x:30, y:170, width:600, height:40)
>
>        // update the game list field frame //
>        GameListField.frame = CGRect(x:230, y:135, width:200, height:20)
>
>        // update the game list font button frame //
>        GameListFontButton.frame = CGRect(x:230, y:110, width:200, height:21)
>
>        // update the game list bold button frame //
>        GameListBoldButton.frame = CGRect(x:230, y:85, width:200, height:21)
>
>        // update the game list font sizes button frame //
>        GameListSizesButton.frame = CGRect(x:230, y:60, width:200, height:21)
>
>        // update the game list color view frame //
>        GameListColorView.frame = CGRect(x:30, y:20, width:600, height:40)
>     }
>
>     // update the stockfish preferences layout //
>     func StockfishPreferencesLayout()
>     {
>        // update the multithread uci engine button frame //
>        MultithreadUCIEngineButton.frame = CGRect(x:30, y:580, width:200, height:20)
>
>        // update the show stockfish analysis button frame //
>        ShowStockfishAnalysisButton.frame = CGRect(x:30, y:530, width:200, height:20)
>
>        // update the uci engine difficulty slider frame //
>        UCIEngineDifficultySlider.frame = CGRect(x:30, y:490, width:600, height:13)
>
>        // update the uci engine difficulty view frame //
>        UCIEngineDifficultyView.frame = CGRect(x:30, y:472, width:600, height:20)
>
>        // update the uci engine hash slider frame //
>        UCIEngineHashSlider.frame = CGRect(x:30, y:440, width:600, height:13)
>
>        // update the uci engine hash view frame //
>        UCIEngineHashView.frame = CGRect(x:30, y:422, width:600, height:20)
>
>        // update the uci engine riskiness slider frame //
>        UCIEngineRiskinessSlider.frame = CGRect(x:30, y:390, width:600, height:13)
>
>        // update the uci engine riskiness view frame //
>        UCIEngineRiskinessView.frame = CGRect(x:30, y:372, width:600, height:20)
>     }
>
>     // the movie preferences layout //
>     func MoviePreferencesLayout()
>     {
>        // update the movie frames per second slider frame //
>        MovieFPSSlider.frame = CGRect(x:30, y:585, width:600, height:13)
>
>        // update the movie frames per second text view frame //
>        MovieFPSView.frame = CGRect(x:30, y:567, width:600, height:20)
>     }
>
>     // set the number of gallery grid rows //
>     @IBAction func SetNumberOfGridRows(_:AnyObject)
>     {
>        // set the number of gallery grid rows //
>        NumberOfGridRows = GridRowsButton.indexOfSelectedItem + 2
>     }
>
>     // the gallery preferences layout //
>     func GalleryPreferencesLayout()
>     {
>        // update the gallery animation rate slider frame //
>        GalleryAnimationRateSlider.frame = CGRect(x:30, y:585, width:600, height:13)
>
>        // update the gallery animation rate view frame //
>        GalleryAnimationRateView.frame = CGRect(x:30, y:567, width:600, height:20)
>
>        // set the grid rows button frame //
>        GridRowsButton.frame = CGRect(x:170, y:510, width:320, height:21)
>     }
>
>     // the diagram preferences layout //
>     func DiagramPreferencesLayout()
>     {
>        // set the display illustrations button frame //
>        DisplayIllustrationsButton.frame = CGRect(x:30, y:580, width:600, height:20)
>
>        // set the illustration size slider frame //
>        IllustrationSizeSlider.frame = CGRect(x:30, y:540, width:600, height:13)
>
>        // set the illustration size view frame //
>        IllustrationSizeView.frame = CGRect(x:30, y:522, width:600, height:20)
>     }
>
>     // update the clock preferences layout //
>     func ClockPreferencesLayout()
>     {
>        // update the display clock button frame //
>        DisplayClockButton.frame = CGRect(x:30, y:580, width:200, height:20)
>
>        // update the game clock color field frame //
>        GameClockColorField.frame = CGRect(x:230, y:540, width:200, height:20)
>
>        // update the clock color view frame //
>        ClockColorView.frame = CGRect(x:30, y:500, width:600, height:40)
>
>        // update the game clock background color field frame //
>        GameClockBackgroundColorField.frame = CGRect(x:230, y:455, width:200, height:20)
>
>        // update the clock background color view frame //
>        ClockBackgroundColorView.frame = CGRect(x:30, y:415, width:600, height:40)
>
>        // update the clock background visibility slider //
>        ClockBackgroundVisibilitySlider.frame = CGRect(x:30, y:385, width:600, height:13)
>
```

```
>       // update the clock background opacity view //
>       ClockBackgroundOpacityView.frame = CGRect(x:30, y:367, width:600, height:20)
>
>       // update the time clock preview //
>       TimeClockPreview.frame = CGRect(x:280, y:245, width:100, height:100)
>    }
>
>    // update the speed preferences layout //
>    func SpeedPreferencesLayout()
>    {
>       // update the animation rate slider frame //
>       AnimationRateSlider.frame = CGRect(x:30, y:585, width:600, height:13)
>
>       // update the animation rate view frame //
>       AnimationRateView.frame = CGRect(x:30, y:567, width:600, height:20)
>
>       // update the search time slider frame //
>       SearchTimeSlider.frame = CGRect(x:30, y:535, width:600, height:13)
>
>       // update the search time view frame //
>       SearchTimeView.frame = CGRect(x:30, y:517, width:600, height:20)
>    }
>
>    // update the gallery preferences state //
>    func GalleryPreferencesToState(_ State:Bool)
>    {
>       // update the grid rows button state //
>       GridRowsButton.isHidden = State
>
>       // update the gallery animation rate slider state //
>       GalleryAnimationRateSlider.isHidden = State
>
>       // update the gallery animation rate view state //
>       GalleryAnimationRateView.isHidden = State
>    }
>
>    // update the diagram preferences state //
>    func DiagramPreferencesToState(_ State:Bool)
>    {
>       // update the display illustrations button state //
>       DisplayIllustrationsButton.isHidden = State
>
>       // update the illustration size slider state //
>       IllustrationSizeSlider.isHidden = State
>
>       // update the illustration size view state //
>       IllustrationSizeView.isHidden = State
>
>       // update the diagram preview state //
>       DiagramPreview.isHidden = State
>    }
>
>    // update the movie preferences state //
>    func MoviePreferencesToState(_ State:Bool)
>    {
>       // update the movie frames per second slider state //
>       MovieFPSSlider.isHidden = State
>
>       // update the movie frames per second view state //
>       MovieFPSView.isHidden = State
>    }
>
>    // update the clock preference state //
>    func ClockPreferencesToState(_ State:Bool)
>    {
>       // update the display clock button state //
>       DisplayClockButton.isHidden = State
>
>       // update the game clock color field state //
>       GameClockColorField.isHidden = State
>
>       // update the clock color view state //
>       ClockColorView.isHidden = State
>
>       // update the game clock background color field state //
>       GameClockBackgroundColorField.isHidden = State
>
>       // update the clock background color view state //
>       ClockBackgroundColorView.isHidden = State
>
>       // update the time clock preview state //
>       TimeClockPreview.isHidden = State
>
>       // update the clock background visibility slider state //
>       ClockBackgroundVisibilitySlider.isHidden = State
>
>       // update the clock background opacity view //
>       ClockBackgroundOpacityView.isHidden = State
>    }
>
>    // update the speed preferences state //
>    func SpeedPreferencesToState(_ State:Bool)
8096,8100c9426,9436
<       // update the game info font //
<       GameInfoFont.UpdateFont()
<
<       // update the game info list //
<       UpdateGameInfoList()
---
>       // update the animation rate slider state //
>       AnimationRateSlider.isHidden = State
>
>       // update the animation rate view state //
>       AnimationRateView.isHidden = State
>
>       // update the search time slider state //
>       SearchTimeSlider.isHidden = State
>
>       // update the search time view state //
>       SearchTimeView.isHidden = State
8103,8104c9439,9442
<    // update the chess piece set type //
<    @IBAction func UpdatePieceSetType(_:AnyObject)
---
>    // update the preference states //
>    func UpdatePreferenceStates(_ Preview:Bool, ClockState:Bool, SpeedState:Bool, StockfishState:
>                                Bool, FontState:Bool, BackgroundState:Bool, BoardState:Bool,
>                                MovieState:Bool, GalleryState:Bool, DiagramState:Bool)
8106,8107c9444,9448
<       // if chess piece set type is different //
<       if PieceSetButton.indexOfSelectedItem != ChessPieceSetType
---
>       // set whether to update the board preview in preferences //
>       UpdatePreview = Preview
>
>       // if need to update the board preview //
>       if UpdatePreview
8109,8110c9450,9451
<          // get the chess piece set type from the piece set button //
<          ChessPieceSetType = PieceSetButton.indexOfSelectedItem
```

```
---
>          // update the board preview image //
>          BoardPreview.image = Graphics.CGImageToNSImage(CGContexts[2].makeImage()!, Size:400)
8112,8113c9453,9458
<          // update the piece set type //
<          PieceImages.UpdatePieceSet(ChessPieceSetType)
---
>          // update the diagram preview image //
>          DiagramPreview.image = BoardPreview.image
>       }
>
>       // update the clock preferences state //
>       ClockPreferencesToState(ClockState)
8115,8116c9460,9473
<          // refresh the board and pieces //
<          RefreshBoardAndPieces(2, UpdateBoard:false, Display:true)
---
>       // update the speed preferences state //
>       SpeedPreferencesToState(SpeedState)
>
>       // update the stockfish preferences state //
>       StockfishPreferencesToState(StockfishState)
>
>       // update the font preferences state //
>       FontPreferencesToState(FontState)
>
>       // update the background preferences state //
>       BackgroundPreferencesToState(BackgroundState)
>
>       // update the board preferences state //
>       BoardPreferencesToState(BoardState)
8118,8120c9475,9482
<          // update the position editor toolbar buttons //
<          UpdatePositionEditorToolbarButtons()
<       }
---
>       // update the movie preferences state //
>       MoviePreferencesToState(MovieState)
>
>       // update the gallery preferences state //
>       GalleryPreferencesToState(GalleryState)
>
>       // update the diagram preferences state //
>       DiagramPreferencesToState(DiagramState)
8123,8124c9485,9486
<    // update the board type //
<    @IBAction func UpdateBoardTypeLayer(_:AnyObject)
---
>    // the clock preferences action //
>    @IBAction func ClockPreferencesAction(_:AnyObject)
8126,8134c9488,9491
<       // if the board type is different //
<       if BoardTexturesButton.indexOfSelectedItem != BoardType
<       {
<          // get the board type from the board textures button //
<          BoardType = BoardTexturesButton.indexOfSelectedItem
<
<          // refresh the board and pieces //
<          RefreshBoardAndPieces(2, UpdateBoard:true, Display:true)
<       }
---
>       // update the preference states //
>       UpdatePreferenceStates(false, ClockState:false, SpeedState:true, StockfishState:true,
>                              FontState:true, BackgroundState:true, BoardState:true, MovieState:true,
>                              GalleryState:true, DiagramState:true)
8137,8138c9494,9495
<    // set the game clock background color //
<    func SetGameClockBackgroundColor()
---
>    // the speed preferences action //
>    @IBAction func SpeedPreferencesAction(_:AnyObject)
8140,8150c9497,9500
<       // set the game clock background color from the clock background color view //
<       GameClockBackgroundHue = ClockBackgroundColorView.GetHueValue()
<
<       // update the white time clock background color //
<       WhiteTimeClock.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
<
<       // update the black time clock background color //
<       BlackTimeClock.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
<
<       // update the time clock preview background color //
<       TimeClockPreview.UpdateClockBackgroundColor(ClockBackgroundColorView.GetCurrentHue())
---
>       // update the preference states //
>       UpdatePreferenceStates(false, ClockState:true, SpeedState:false, StockfishState:true,
>                              FontState:true, BackgroundState:true, BoardState:true, MovieState:true,
>                              GalleryState:true, DiagramState:true)
8153,8154c9503,9504
<    // set the game clock color //
<    func SetGameClockColor()
---
>    // the stockfish preferences action //
>    @IBAction func StockfishPreferencesAction(_:AnyObject)
8156,8166c9506,9509
<       // set the game clock color from the clock color view //
<       GameClockHue = ClockColorView.GetHueValue()
<
<       // update the white time clock border color //
<       WhiteTimeClock.UpdateClockBorderColor(ClockColorView.GetCurrentHue())
<
<       // update the black time clock border color //
<       BlackTimeClock.UpdateClockBorderColor(ClockColorView.GetCurrentHue())
<
<       // update the time clock border preview color //
<       TimeClockPreview.UpdateClockBorderColor(ClockColorView.GetCurrentHue())
---
>       // update the preference states //
>       UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:false,
>                              FontState:true, BackgroundState:true, BoardState:true, MovieState:true,
>                              GalleryState:true, DiagramState:true)
8169,8170c9512,9513
<    // initialize the preference layouts //
<    func InitializePreferenceLayouts()
---
```

```
>     // the gallery preferences action //
>     @IBAction func GalleryPreferencesAction(_:AnyObject)
8172,8257c9515,9518
<         // set the board controls list //
<         BoardControlsList = [BoardPreview, BoardTexturesButton, PieceSetButton, AutosaveGameButton,
<                              GameSoundsButton, EcoBookButton, HighlightLastMoveMadeButton,
<                              HighlightCapturedPiecesButton, HighlightThreatenedPiecesButton,
<                              MoveBySelectingSquaresButton, DisplayCapturedPiecesButton,
<                              RotateBoardButton, BorderTextureView, BorderTextureSlider,
<                              BorderVisibilityView, BorderVisibilitySlider, SquareBorderView,
<                              SquareBorderOpacitySlider, BoardRoughnessView, BoardRoughnessSlider,
<                              BoardTintColorView, BoardTintColorSlider, BoardTintVisibilityView,
<                              BoardTintVisibilitySlider]
<
<         // initialize the board controls //
<         BoardControls.Initialize(BoardControlsList, Frames:[[30, 200, 400, 400], [444, 579, 165, 21],
<                                  [444, 540, 165, 21], [445, 501, 200, 20], [445, 463, 200, 20],
<                                  [445, 425, 200, 20], [445, 387, 200, 20], [445, 349, 200, 20],
<                                  [445, 311, 200, 20], [445, 273, 200, 20], [445, 235, 200, 20],
<                                  [445, 197, 200, 20], [445, 149, 200, 20], [30, 152, 400, 13],
<                                  [445, 123, 200, 20], [30, 126, 400, 13], [445, 97, 200, 20],
<                                  [30, 100, 400, 13], [445, 71, 200, 20], [30, 74, 400, 13],
<                                  [445, 45, 200, 20], [30, 48, 400, 13], [445, 19, 200, 20],
<                                  [30, 22, 400, 13]])
<
<         // set the font controls list //
<         FontControlsList = [BoardLabelsField, BoardLabelsFontButton, BoardLabelsBoldButton,
<                             BoardLabelsSizesButton, BoardLabelsColorView, GameInfoField,
<                             GameInfoFontButton, GameInfoBoldButton, GameInfoSizesButton,
<                             GameInfoColorView, MoveListField, MoveListFontButton, MoveListBoldButton,
<                             MoveListSizesButton, MoveListColorView, GameListField, GameListFontButton,
<                             GameListBoldButton, GameListSizesButton, GameListColorView]
<
<         // initialize the font controls //
<         FontControls.Initialize(FontControlsList, Frames:[[230, 585, 200, 20], [230, 560, 200, 21],
<                                 [230, 535, 200, 21], [230, 510, 200, 21], [30, 470, 600, 40],
<                                 [230, 435, 200, 20], [230, 410, 200, 21], [230, 385, 200, 21],
<                                 [230, 360, 200, 21], [30, 320, 600, 40], [230, 285, 200, 20],
<                                 [230, 260, 200, 21], [230, 235, 200, 21], [230, 210, 200, 21],
<                                 [30, 170, 600, 40], [230, 135, 200, 20], [230, 110, 200, 21],
<                                 [230, 85, 200, 21], [230, 60, 200, 21], [30, 20, 600, 40]])
<
<         // initialize the clock controls //
<         ClockControls.Initialize([DisplayClockButton, GameClockColorField, ClockColorView,
<                                  GameClockBackgroundColorField, ClockBackgroundColorView,
<                                  ClockBackgroundVisibilitySlider, ClockBackgroundOpacityView,
<                                  TimeClockPreview], Frames:[[30, 580, 200, 20], [230, 540, 200, 20],
<                                  [30, 500, 600, 40], [230, 455, 200, 20], [30, 415, 600, 40],
<                                  [30, 385, 600, 13], [30, 367, 600, 20], [280, 245, 100, 100]])
<
<         // initialize the stockfish controls //
<         StockfishControls.Initialize([MultithreadUCIEngineButton, ShowStockfishAnalysisButton,
<                                      UCIEngineDifficultySlider, UCIEngineDifficultyView,
<                                      UCIEngineHashSlider, UCIEngineHashView, UCIEngineRiskinessSlider,
<                                      UCIEngineRiskinessView], Frames:[[30, 580, 200, 20],
<                                      [30, 530, 200, 20], [30, 490, 600, 13], [30, 472, 600, 20],
<                                      [30, 440, 600, 13], [30, 422, 600, 20], [30, 390, 600, 13],
<                                      [30, 372, 600, 20]])
<
<         // initialize the background controls //
<         BackgroundControls.Initialize([WindowTextureOpacitySlider, WindowTextureOpacityView,
<                                       WindowColorsBackgroundField, WindowBackgroundColorView,
<                                       WindowColorPreview], Frames:[[30, 585, 600, 13],
<                                       [30, 567, 600, 20], [230, 540, 200, 20], [30, 500, 600, 40],
<                                       [230, 270, 200, 200]])
<
<         // initialize the speed controls //
<         SpeedControls.Initialize([AnimationRateSlider, AnimationRateView, SearchTimeSlider,
<                                  SearchTimeView], Frames:[[30, 585, 600, 13], [30, 567, 600, 20],
<                                  [30, 535, 600, 13], [30, 517, 600, 20]])
<
<         // initialize the diagram controls //
<         DiagramControls.Initialize([DisplayIllustrationsButton, IllustrationSizeSlider,
<                                    IllustrationSizeView, DiagramPreview], Frames:[[30, 580, 600, 20],
<                                    [30, 540, 600, 13], [30, 522, 600, 20]])
<
<         // initialize the gallery controls //
<         GalleryControls.Initialize([GalleryAnimationRateSlider, GalleryAnimationRateView,
<                                    GridRowsButton], Frames:[[30, 585, 600, 13], [30, 567, 600, 20],
<                                    [170, 510, 320, 21]])
<
<         // initialize the movie controls //
<         MovieControls.Initialize([MovieFPSSlider, MovieFPSView], Frames:[[30, 585, 600, 13],
<                                  [30, 567, 600, 20]])
<
<         // set the preference controls dictionary //
<         PrefControls = [20:BackgroundControls, 21:BoardControls, 22:FontControls,
<                         23:StockfishControls, 24:SpeedControls, 25:ClockControls, 26:MovieControls,
<                         27:GalleryControls, 28:DiagramControls]
---
>         // update the preference states //
>         UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
>                                FontState:true, BackgroundState:true, BoardState:true, MovieState:true,
>                                GalleryState:false, DiagramState:true)
8260,8261c9521,9522
<     // set the number of gallery grid rows //
<     @IBAction func SetNumberOfGridRows(_:AnyObject)
---
>     // the diagram preferences action //
>     @IBAction func DiagramPreferencesAction(_:AnyObject)
8263,8264c9524,9527
<         // set the number of gallery grid rows //
<         NumberOfGridRows = GridRowsButton.indexOfSelectedItem + 2
---
>         // update the preference states //
>         UpdatePreferenceStates(true, ClockState:true, SpeedState:true, StockfishState:true,
>                                FontState:true, BackgroundState:true, BoardState:true, MovieState:true,
>                                GalleryState:true, DiagramState:false)
8267,8268c9530,9531
<     // the preferences window action //
<     @IBAction func PreferencesAction(_:AnyObject)
---
>     // the movie preferences action //
>     @IBAction func MoviePreferencesAction(_:AnyObject)
8270,8271c9533,9537
<         // update the board preview image //
<         BoardPreview.image = Graphics.CGImageToNSImage(CGContexts[2].makeImage()!, Size:400)
---
>         // update the preference states //
>         UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
>                               FontState:true, BackgroundState:true, BoardState:true, MovieState:
>                               false, GalleryState:true, DiagramState:true)
>     }
8273,8274c9539,9546
<         // update the diagram preview image //
<         DiagramPreview.image = BoardPreview.image
---
```

```
>    // the board preferences action //
>    @IBAction func BoardPreferencesAction(_:AnyObject)
>    {
>       // update the preference states //
>       UpdatePreferenceStates(true, ClockState:true, SpeedState:true, StockfishState:true,
>                             FontState:true, BackgroundState:true, BoardState:false, MovieState:
>                             true, GalleryState:true, DiagramState:true)
>    }
8276,8314c9548,9563
<       // get the current preferences selection //
<       PreferencesSelection = Int(PreferencesToolbar.selectedItemIdentifier!.rawValue)!
<
<       // if the preferences selection is the same as the previous //
<       if PreferencesSelection == PreviousPreferencesSelection
<       {
<          // exit since the preferences selection hasn't changed //
<          return
<       }
<
<       // if the preferences selection is for the board preferences //
<       if PreferencesSelection == 21
<       {
<          // set the update the board preview //
<          UpdatePreview = true
<       }
<       // not for the board preferences //
<       else
<       {
<          // set that don't need to update the board preview //
<          UpdatePreview = false
<       }
<
<       // if the selected preference tab exists in the preference controls dictionary //
<       if PrefControls[PreferencesSelection] != nil
<       {
<          // if the previously selected preference tab exists in the pref controls dictionary //
<          if PrefControls[PreviousPreferencesSelection] != nil
<          {
<             // hide the previous preferences selection //
<             PrefControls[PreviousPreferencesSelection]!.UpdateState(true)
<          }
<
<          // display the current preferences selection //
<          PrefControls[PreferencesSelection]!.UpdateState(false)
<
<          // save the previous preferences selection //
<          PreviousPreferencesSelection = PreferencesSelection
<       }
---
>    // the background preferences states action //
>    @IBAction func BackgroundPreferencesAction(_:AnyObject)
>    {
>       // update the preference states //
>       UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
>                             FontState:true, BackgroundState:false, BoardState:true, MovieState:
>                             true, GalleryState:true, DiagramState:true)
>    }
>
>    // the font preferences action //
>    @IBAction func FontPreferencesAction(_:AnyObject)
>    {
>       // update the preference states //
>       UpdatePreferenceStates(false, ClockState:true, SpeedState:true, StockfishState:true,
>                             FontState:false, BackgroundState:true, BoardState:true, MovieState:
>                             true, GalleryState:true, DiagramState:true)
8455,8457d9703
<
<       // update the position editor toolbar buttons //
<       UpdatePositionEditorToolbarButtons()
8642c9888
<          Prefs.Save(SavedPositionSlot, Key:"SavedPositionSlot")
---
>          AppPreferences.set(SavedPositionSlot, forKey:"SavedPositionSlot")
9137c10383
<       GameExplorerTabs.UpdateState(true)
---
>       SetGameExplorerButtons(true)
9979c11225
<       HumanOpponentMenuItem.state = NSControl.StateValue(rawValue:BoolTo().Num(HumanOpponent))
---
>       HumanOpponentMenuItem.state = NSControl.StateValue(rawValue:BoolToInt(HumanOpponent))
9982c11228
<       EngineOpponentMenuItem.state = NSControl.StateValue(rawValue:BoolTo().Num(EngineOpponent))
---
>       EngineOpponentMenuItem.state = NSControl.StateValue(rawValue:BoolToInt(EngineOpponent))
9985,9986c11231
<       UCIEngineOpponentMenuItem.state = NSControl.StateValue(rawValue:BoolTo()
<                                                              .Num(UCIEngineOpponent))
---
>       UCIEngineOpponentMenuItem.state = NSControl.StateValue(rawValue:BoolToInt(UCIEngineOpponent))
9992c11237
<       Prefs.Save(Opponent, Key:"Opponent")
---
>       AppPreferences.set(Opponent, forKey:"Opponent")
10350c11595,11597
<       GameListData.RefreshGameList(GameListFont)
---
>       GameListData.RefreshGameList(GameListColorView.GetCurrentHue(), TheFont:
>                                   GetFont(GameListFontType, Size:CGFloat(16 + GameListSizes),
>                                   Bold:GameListBold))
10370c11617,11619
<       GameInfoData.RefreshGameInfoList(GameInfoList, GameInfoFont:GameInfoFont)
---
>       GameInfoData.RefreshGameInfoList(GameInfoList, Color:GameInfoColorView.GetCurrentHue(), Font:
>                                       GetFont(GameInfoFontType, Size:CGFloat(16 + GameInfoSizes),
>                                       Bold:GameInfoBold))
10382,10383c11631,11634
<       // refresh the pgn moves list //
<       PGNMovesData.RefreshPGNMovesList(PGNMovesList, MoveListFont:MoveListFont)
---
```

```
>        // refresh the pgn move list //
>        PGNMovesData.RefreshPGNMovesList(PGNMovesList, Color:MoveListColorView.GetCurrentHue(), Font:
>                                        GetFont(MoveListFontType, Size:CGFloat(16 + MoveListSizes),
>                                        Bold:MoveListBold))
10477a11729,11779
>     // convert an integer to a boolean //
>     func IntToBool(_ IntVar:Int) -> Bool
>     {
>        // if integer is zero //
>        if IntVar == 0
>        {
>           // return false //
>           return false
>        }
>        // integer is not zero //
>        else
>        {
>           // return true //
>           return true
>        }
>     }
>
>     // convert a boolean to a cgfloat //
>     func BoolToCGFloat(_ BoolVar:Bool) -> CGFloat
>     {
>        // if boolean is true //
>        if BoolVar
>        {
>           // return one //
>           return 1
>        }
>        // boolean is false //
>        else
>        {
>           // return zero //
>           return 0
>        }
>     }
>
>     // convert a boolean to an integer //
>     func BoolToInt(_ BoolVar:Bool) -> Int
>     {
>        // if boolean is true //
>        if BoolVar
>        {
>           // return one //
>           return 1
>        }
>        // boolean is false //
>        else
>        {
>           // return zero //
>           return 0
>        }
>     }
>
10550,10552c11852,11854
<
<     // the position pieces action for the position editor //
<     @IBAction func PositionPiecesAction(_:AnyObject)
---
>
>     // the position editor white pawn action //
>     @IBAction func PositionEditorWhitePawnAction(_:AnyObject)
10554,10562c11856,11970
<        // if the selected items is in the position pieces dictionary //
<        if PositionPieces[MainToolbar!.selectedItemIdentifier!.rawValue] != nil
<        {
<           // play the game sound of a click //
<           PlayGameSound("click")
<
<           // set the position cursor to the current piece type //
<           SetPosCursor(PositionPieces[MainToolbar!.selectedItemIdentifier!.rawValue]!)
<        }
---
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a white pawn //
>        SetPosCursor(1)
>     }
>
>     // the position editor white knight action //
>     @IBAction func PositionEditorWhiteKnightAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a white knight //
>        SetPosCursor(2)
>     }
>
>     // the position editor white bishop action //
>     @IBAction func PositionEditorWhiteBishopAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a white bishop //
>        SetPosCursor(3)
>     }
>
>     // the position editor white rook action //
>     @IBAction func PositionEditorWhiteRookAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a white rook //
>        SetPosCursor(4)
>     }
>
>     // the position editor white queen action //
>     @IBAction func PositionEditorWhiteQueenAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a white queen //
>        SetPosCursor(5)
>     }
>
>     // the position editor white king action //
>     @IBAction func PositionEditorWhiteKingAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a white king //
>        SetPosCursor(6)
>     }
>
```

```
>     // the position editor black pawn action //
>     @IBAction func PositionEditorBlackPawnAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a black pawn //
>        SetPosCursor(-1)
>     }
>
>     // the position editor black knight action //
>     @IBAction func PositionEditorBlackKnightAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a black knight //
>        SetPosCursor(-2)
>     }
>
>     // the position editor black bishop action //
>     @IBAction func PositionEditorBlackBishopAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a black bishop //
>        SetPosCursor(-3)
>     }
>
>     // the position editor black rook action //
>     @IBAction func PositionEditorBlackRookAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a black rook //
>        SetPosCursor(-4)
>     }
>
>     // the position editor black queen action //
>     @IBAction func PositionEditorBlackQueenAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a black queen //
>        SetPosCursor(-5)
>     }
>
>     // the position editor black king action //
>     @IBAction func PositionEditorBlackKingAction(_:AnyObject)
>     {
>        // play the game sound of a click //
>        PlayGameSound("click")
>
>        // set the cursor image to a black king //
>        SetPosCursor(-6)
10896a12305,12366
>     // set the gallery button states //
>     func SetGalleryButtonsToState(_ State:Bool)
>     {
>        // update the game gallery left button state //
>        GameGalleryLeft.isEnabled = State
>
>        // update the game gallery right button state //
>        GameGalleryRight.isEnabled = State
>
>        // update the game gallery movie button state //
>        GameGalleryMovie.isEnabled = State
>
>        // update the game gallery snapshot button state //
>        GameGallerySnapshot.isEnabled = State
>
>        // update the game gallery erase button state //
>        GameGalleryErase.isEnabled = State
>
>        // update the game gallery blue color button state //
>        GameGalleryBlueColor.isEnabled = State
>
>        // update the game gallery cyan color button state //
>        GameGalleryCyanColor.isEnabled = State
>
>        // update the game gallery magenta color button state //
>        GameGalleryMagentaColor.isEnabled = State
>
>        // update the game gallery green color button state //
>        GameGalleryGreenColor.isEnabled = State
>
>
>        // update the game gallery red color button state //
>        GameGalleryRedColor.isEnabled = State
>
>        // update the game gallery yellow color button state //
>        GameGalleryYellowColor.isEnabled = State
>
>        // update the game gallery black color button state //
>        GameGalleryBlackColor.isEnabled = State
>
>        // update the game gallery red color button state //
>        GameGalleryWhiteColor.isEnabled = State
>
>        // update the game gallery undo path button state //
>        GameGalleryUndoPath.isEnabled = State
>
>        // update the gallery slider state //
>        GameGallerySlider.isEnabled = State
>
>        // if reviewing //
>        if GameGalleryView.Reviewing
>        {
>           // set the gallery review button icon to the pause icon //
>           GameGalleryReviewButton.image = PauseIcon
>        }
>        // if not reviewing //
>        else
>        {
>           // set the gallery review button icon to the play icon //
>           GameGalleryReviewButton.image = PlayIcon
>        }
>     }
>
10946c12416
<        GalleryTabs.UpdateState(!GameGalleryView.Reviewing)
---
>        SetGalleryButtonsToState(!GameGalleryView.Reviewing)
10973c12443,12446
<        GameExplorerTabs.UpdateState(false)
---
>        SetGameExplorerButtons(false)
>
>        // enable the game review button //
>        GameReviewButton.isEnabled = true
11049,11060d12521
<
<        // loop over all the toolbar items //
<        for Index in 0 ..< MainToolbar.items.count
<        {
<           // if the toolbar item is in the position pieces dictionary //
<           if PositionPieces[MainToolbar.items[Index].itemIdentifier.rawValue] != nil
<           {
<              // update the item image to the correct piece set and piece type //
<              MainToolbar.items[Index].image = PieceImages.GetPieceNSImage(PositionPieces[MainToolbar
<                                              .items[Index].itemIdentifier.rawValue]!)
<           }
<        }
11073a12535,12570
>        // update the position editor white pawn image //
>        PositionEditorWhitePawn.image = PieceImages.GetPieceNSImage(1)
>
>        // update the position editor white knight image //
>        PositionEditorWhiteKnight.image = PieceImages.GetPieceNSImage(2)
>
>        // update the position editor white bishop image //
>        PositionEditorWhiteBishop.image = PieceImages.GetPieceNSImage(3)
>
>        // update the position editor white rook image //
>        PositionEditorWhiteRook.image = PieceImages.GetPieceNSImage(4)
>
>        // update the position editor white queen image //
>        PositionEditorWhiteQueen.image = PieceImages.GetPieceNSImage(5)
>
>        // update the position editor white king image //
>        PositionEditorWhiteKing.image = PieceImages.GetPieceNSImage(6)
>
>        // update the position editor black pawn image //
>        PositionEditorBlackPawn.image = PieceImages.GetPieceNSImage(-1)
>
>        // update the position editor black knight image //
>        PositionEditorBlackKnight.image = PieceImages.GetPieceNSImage(-2)
>
>        // update the position editor black bishop image //
>        PositionEditorBlackBishop.image = PieceImages.GetPieceNSImage(-3)
>
>        // update the position editor black rook image //
>        PositionEditorBlackRook.image = PieceImages.GetPieceNSImage(-4)
>
>        // update the position editor black queen image //
>        PositionEditorBlackQueen.image = PieceImages.GetPieceNSImage(-5)
>
>        // update the position editor black king image //
>        PositionEditorBlackKing.image = PieceImages.GetPieceNSImage(-6)
>
11237c12734
<           if PromoteToDictionary[ReturnCode] != nil
---
>           if PromoteToDictionary["\(ReturnCode)"] != nil
11240c12737
<              PromotionSymbol = PromoteToDictionary[ReturnCode]!
---
>              PromotionSymbol = PromoteToDictionary["\(ReturnCode)"]!
11275c12772
<           Prefs.Save(AutoloadStockfish, Key:"AutoloadStockfish")
---
>           AppPreferences.set(AutoloadStockfish, forKey:"AutoloadStockfish")
11283,11285c12780,12782
<        RankAndFile.Set(abs(BoolTo().Num(RotateBoard) * 7 - Int((MouseLocation.x - Dims
<                        .GetBorderWidth()) / Dims.GetSquareSize(2))), Y:abs(BoolTo().Num(RotateBoard)
<                        * 7 - Int((MouseLocation.y - Dims.GetBorderWidth()) / Dims.GetSquareSize(2))))
---
>        RankAndFile.Set(abs(BoolToInt(RotateBoard) * 7 - Int((MouseLocation.x - Dims.GetBorderWidth())
>                        / Dims.GetSquareSize(2))), Y:abs(BoolToInt(RotateBoard) * 7 -
>                        Int((MouseLocation.y - Dims.GetBorderWidth()) / Dims.GetSquareSize(2))))
```