```
2c2
< // Swift source code of main.swift Version 8.1.0 at greenchess.com //
---
> // Swift source code of main.swift Version 8.0.8 at greenchess.com //
354c354
<    var X:CGFloat = 0
---
>    var X:Float = 0
357c357
<    var Y:CGFloat = 0
---
>    var Y:Float = 0
360c360
<    var Angle:CGFloat = 0
---
>    var Angle:Float = 0
363c363
<    var WhiteTime:CGFloat = 0
---
>    var WhiteTime:Float = 0
366c366
<    var BlackTime:CGFloat = 0
---
>    var BlackTime:Float = 0
402c402
<       WhiteTime = CGFloat(ElapsedWhiteTime)
---
>       WhiteTime = ElapsedWhiteTime
405c405
<       BlackTime = CGFloat(ElapsedBlackTime)
---
>       BlackTime = ElapsedBlackTime
446c446
<       HourHands.lineWidth = 5
---
>       HourHands.lineWidth = 6
458c458
<       ClockBorder.lineWidth = 2
---
>       ClockBorder.lineWidth = 3
461c461
<       ClockBorder.appendOval(in:NSMakeRect(2, 2, 100, 100))
---
>       ClockBorder.appendOval(in:NSMakeRect(3, 3, 125, 125))
464c464
<       ClockBorder.appendOval(in:NSMakeRect(122, 2, 100, 100))
---
>       ClockBorder.appendOval(in:NSMakeRect(183, 3, 125, 125))
470c470
<       ClockFace.appendOval(in:NSMakeRect(48, 48, 8, 8))
---
>       ClockFace.appendOval(in:NSMakeRect(60.5, 60.5, 10, 10))
473c473
<       ClockFace.appendOval(in:NSMakeRect(168, 48, 8, 8))
---
>       ClockFace.appendOval(in:NSMakeRect(240.5, 60.5, 10, 10))
475,476c475,476
<       // loop over twelve tick marks //
<       for Index in 0 ..< 12
---
>       // loop over sixty tick marks //
>       for Index in 0 ..< 60
479c479
<          Angle = CGFloat(Index) * CGFloat.pi / 6
---
>          Angle = Float(Index) * Float.pi / 30
482c482
<          X = 51 + 47 * sin(Angle)
---
>          X = 65 + 60 * sin(Angle)
485c485,491
<          Y = 51 + 47 * cos(Angle)
---
>          Y = 65 + 60 * cos(Angle)
>
>          // if currently on an hour tick mark //
>          if Index % 5 == 0
>          {
>             // make an hour tick mark for the white clock //
>             ClockFace.appendOval(in:NSMakeRect(CGFloat(X - 1), CGFloat(Y - 1), 3, 3))
487,488c493,500
<          // make an hour tick mark for the white clock //
<          ClockFace.appendOval(in:NSMakeRect(X, Y, 2, 2))
---
>             // make an hour tick mark for the black clock //
>             ClockFace.appendOval(in:NSMakeRect(CGFloat(X + 179), CGFloat(Y - 1), 3, 3))
>          }
>          // if currently on a minute tick mark //
>          else
>          {
>             // make a minute tick mark for the white clock //
>             ClockFace.appendOval(in:NSMakeRect(CGFloat(X), CGFloat(Y), 1, 1))
490,491c502,504
<          // make an hour tick mark for the black clock //
<          ClockFace.appendOval(in:NSMakeRect(X + 120, Y, 2, 2))
---
>             // make a minute tick mark for the black clock //
>             ClockFace.appendOval(in:NSMakeRect(CGFloat(X + 180), CGFloat(Y), 1, 1))
>          }
535c548
<       SecondHands.move(to:NSMakePoint(52, 52))
---
>       SecondHands.move(to:NSMakePoint(65.5, 65.5))
538c551
<       X = 52 + 48 * sin(CGFloat.pi * WhiteTime / 30)
---
>       X = 65.5 + 60 * sin(Float.pi * WhiteTime / 30)
541c554
<       Y = 52 + 48 * cos(CGFloat.pi * WhiteTime / 30)
---
>       Y = 65.5 + 60 * cos(Float.pi * WhiteTime / 30)
544c557
<       SecondHands.line(to:NSMakePoint(X, Y))
---
>       SecondHands.line(to:NSMakePoint(CGFloat(X), CGFloat(Y)))
547c560
<       SecondHands.move(to:NSMakePoint(172, 52))
---
>       SecondHands.move(to:NSMakePoint(245.5, 65.5))
550c563
<       X = 172 + 48 * sin(CGFloat.pi * BlackTime / 30)
---
>       X = 245.5 + 60 * sin(Float.pi * BlackTime / 30)
553c566
<       Y = 52 + 48 * cos(CGFloat.pi * BlackTime / 30)
---
>       Y = 65.5 + 60 * cos(CGFloat.pi * BlackTime / 30)
556c569
<       SecondHands.line(to:NSMakePoint(X, Y))
---
>       SecondHands.line(to:NSMakePoint(CGFloat(X), CGFloat(Y)))
562c575
<       MinuteHands.move(to:NSMakePoint(52, 52))
---
>       MinuteHands.move(to:NSMakePoint(65.5, 65.5))
565c578
<       X = 52 + 36 * sin(CGFloat.pi * WhiteTime / 1800)
---
>       X = 65.5 + 50 * sin(Float.pi * WhiteTime / 1800)
568c581
<       Y = 52 + 36 * cos(CGFloat.pi * WhiteTime / 1800)
---
>       Y = 65.5 + 50 * cos(Float.pi * WhiteTime / 1800)
571c584
<       MinuteHands.line(to:NSMakePoint(X, Y))
---
>       MinuteHands.line(to:NSMakePoint(CGFloat(X), CGFloat(Y)))
574c587
<       MinuteHands.move(to:NSMakePoint(172, 52))
---
>       MinuteHands.move(to:NSMakePoint(245.5, 65.5))
577c590
<       X = 172 + 36 * sin(CGFloat.pi * BlackTime / 1800)
---
>       X = 245.5 + 50 * sin(Float.pi * BlackTime / 1800)
580c593
<       Y = 52 + 36 * cos(CGFloat.pi * BlackTime / 1800)
---
>       Y = 65.5 + 50 * cos(Float.pi * BlackTime / 1800)
583c596
<       MinuteHands.line(to:NSMakePoint(X, Y))
---
>       MinuteHands.line(to:NSMakePoint(CGFloat(X), CGFloat(Y)))
589c602
<       HourHands.move(to:NSMakePoint(52, 52))
---
>       HourHands.move(to:NSMakePoint(65.5, 65.5))
592c605
<       X = 52 + 24 * sin(CGFloat.pi * WhiteTime / 21600)
---
>       X = 65.5 + 30 * sin(Float.pi * WhiteTime / 21600)
595c608
<       Y = 52 + 24 * cos(CGFloat.pi * WhiteTime / 21600)
---
>       Y = 65.5 + 30 * cos(Float.pi * WhiteTime / 21600)
598c611
<       HourHands.line(to:NSMakePoint(X, Y))
---
>       HourHands.line(to:NSMakePoint(CGFloat(X), CGFloat(Y)))
601c614
<       HourHands.move(to:NSMakePoint(172, 52))
---
>       HourHands.move(to:NSMakePoint(245.5, 65.5))
604c617
<       X = 172 + 24 * sin(CGFloat.pi * BlackTime / 21600)
---
>       X = 245.5 + 30 * sin(Float.pi * BlackTime / 21600)
607c620
<       Y = 52 + 24 * cos(CGFloat.pi * BlackTime / 21600)
---
>       Y = 65.5 + 30 * cos(Float.pi * BlackTime / 21600)
610c623
<       HourHands.line(to:NSMakePoint(X, Y))
---
>       HourHands.line(to:NSMakePoint(CGFloat(X), CGFloat(Y)))
616a630,729
> // the captured pieces display interface //
> class CapturedPiecesDisplay:NSView
> {
>    // the type of chess piece set to use //
>    var ChessPieceSetType:Int = 0
>
>    // the chess piece set type image index //
>    var ChessPieceImageIndex:Int = 0
>
>    // an index in a list //
>    var Index:Int = 0
>
>    // the number of captured pieces displayed to the view so far //
>    var CapturedPieceCount:Int = 0
>
>    // the list of captured pieces to display //
>    var CapturedPiecesToDisplay:[Int] = [0, 0, 0, 0, 0]
>
>    // whether the piece is white or not //
>    var WhitePiece:Bool = false
>
>    // the image of a piece //
>    var PieceImage:NSImage = NSImage()
>
>    // an array of chess piece images //
>    var ChessPieceImages:[NSImage] = []
>
>    // the piece width //
>    var Width:Int = 0
>
>    // set the chess piece set images //
>    func SetChessPieceSetImages(_ ChessPieceSetImages:[NSImage])
>    {
>       // set the chess piece images //
>       ChessPieceImages = ChessPieceSetImages
>    }
>
>    // update the captured pieces display //
>    func UpdateCapturedPiecesDisplay(_ CapturedPieces:[Int], White:Bool, Set:Int)
>    {
>       // set the chess piece set type //
>       ChessPieceSetType = Set
>
>       // set whether captured pieces are white or not //
>       WhitePiece = White
>
>       // copy over the captured pieces //
>       CapturedPiecesToDisplay = CapturedPieces
>
>       // update the captured pieces display //
>       displayIgnoringOpacity(bounds)
>    }
>
>    // set the piece image for the given piece type //
>    func SetPieceImage(_ PieceType:Int)
>    {
>       // if the piece is white //
>       if WhitePiece
>       {
>          // set the chess piece image index corresponding to color and piece set type //
>          ChessPieceImageIndex = abs(PieceType - 4) + ChessPieceSetType * 12
>       }
>       // piece is black //
>       else
>       {
>          // set the chess piece image index corresponding to color and piece set type //
>          ChessPieceImageIndex = abs(PieceType - 4) + (ChessPieceSetType * 12 + 6)
>       }
>
>       // set the chess piece image //
>       PieceImage = ChessPieceImages[ChessPieceImageIndex]
>    }
>
>    // redefine the draw rectangle function //
>    override func draw(_ Rect:NSRect)
>    {
>       // initialize the captured pieces count //
>       CapturedPieceCount = 0
>
>       // loop over the list of captured pieces to display //
>       for Index in 0 ..< 5
>       {
>          // set the piece image for the captured piece //
>          SetPieceImage(Index)
>
>          // loop over the number of pieces captures for this particular piece type //
>          for _ in 0 ..< CapturedPiecesToDisplay[Index]
>          {
>             // draw the piece in the view //
>             PieceImage.draw(in:NSMakeRect(CGFloat(CapturedPieceCount * Width), 0, CGFloat(Width),
>                         CGFloat(Width)), from:NSZeroRect, operation:NSCompositingOperation
>                         .sourceOver, fraction:1)
>
>             // increment the captured piece count //
>             CapturedPieceCount += 1
>          }
>       }
>    }
> }
>
623c736
<    // override the key down method //
---
>    // override the key down method
1053c1166
<
---
>
1907,1912d2019
<    // the number of captured pieces to display in the board //
<    var CapturedPieceCount:Int = 0
<
<    // the movie info tag position on the movie info frame //
<    var MovieInfoTagPosition:CGFloat = 0
<
1991,1993d2097
<    // the movie info layer for the game movie //
<    var MovieInfoLayer:CGLayer?
<
1997,2002d2100
<    // rectangles in which to draw the captured pieces //
<    var CRects:[CGRect] = []
<
<    // rectangle in which to draw the background for the captured pieces //
<    var CRectBackground:CGRect = CGRect(x:0, y:0, width:1, height:1)
<
2153,2154c2251,2255
<    // whether to hide the captured pieces in the board or not //
<    var HideCapturedPiecesInBoard:Bool = false
---
>    // whether to hide the captured pieces in the panel or not //
>    var HideCapturedPiecesInPanel:Bool = false
>
>    // whether to hide the captured pieces in the panel or not //
>    var HidePieces:CGFloat = 0
2224,2226d2324
<
<    // the movie info attributes //
<    var MovieInfoAttributes:NSDictionary = [:]
2321,2323d2418
<    // the attributed movie tag string //
<    var AttributedMovieTag = NSAttributedString()
<
2957c3052
<
---
>
3558,3559c3653,3654
<    // the hide captured pieces in board menu item //
<    @IBOutlet var HideCapturedPiecesInBoardMenuItem:NSMenuItem!
---
>    // the hide captured pieces in panel menu item //
>    @IBOutlet var HideCapturedPiecesInPanelMenuItem:NSMenuItem!
3874a3970,3975
>
>    // the view to display the captured white pieces on //
>    @IBOutlet var CapturedPiecesWhiteView:CapturedPiecesDisplay!
>
>    // the view to display the captured black pieces on //
>    @IBOutlet var CapturedPiecesBlackView:CapturedPiecesDisplay!
3923,3924c4024,4025
<       // create a game timer that gets called five times per second //
<       RunLoop.current.add(Timer(timeInterval:0.2, target:self, selector:#selector(GreenChess
---
>       // create a game timer that gets called ten times per second //
>       RunLoop.current.add(Timer(timeInterval:0.1, target:self, selector:#selector(GreenChess
4325c4426
<             return min(CGFloat(PGNMovesData.RowHeight(TableRow / 2)), CGBoardSize[2])
---
>             return min(CGFloat(PGNMovesData.RowHeight(TableRow / 2)), 660)
4915,4916c5016,5017
<    // hide the captured pieces in the board menu item action //
<    @IBAction func HideCapturedPiecesInBoardMenuItemAction(_:AnyObject)
---
>    // hide the captured pieces in the panel menu item action //
>    @IBAction func HideCapturedPiecesInPanelMenuItemAction(_:AnyObject)
4918c5019
<       // add a timer to run the hide captured pieces in board menu item action //
---
>       // add a timer the run the hide captured pieces in panel menu item action //
4920c5021
<                      .TheHideCapturedPiecesInBoardMenuItemAction), userInfo:nil, repeats:
---
>                      .TheHideCapturedPiecesInPanelMenuItemAction), userInfo:nil, repeats:
4924,4925c5025,5026
<    // the hide captured pieces in board menu item action //
<    @objc func TheHideCapturedPiecesInBoardMenuItemAction()
---
>    // the hide captured pieces in panel menu item action //
>    @objc func TheHideCapturedPiecesInPanelMenuItemAction()
4927,4931c5028,5029
<       // toggle the hide captured pieces in board //
<       HideCapturedPiecesInBoard = !HideCapturedPiecesInBoard
<
<       // update the board and captured pieces //
<       UpdateBoardAndCapturedPieces()
---
>       // toggle the hide captured pieces in panel //
>       HideCapturedPiecesInPanel = !HideCapturedPiecesInPanel
4933,4934c5031,5032
<       // update the hide captured pieces in board preference //
<       AppPreferences.set(HideCapturedPiecesInBoard, forKey:"HideCapturedPiecesInBoard")
---
>       // hide or unhide the captured pieces in the panel //
>       HideCapturedPiecesInPanelAction()
4936,4938c5034,5035
<       // update the hide captured pieces in board menu item state //
<       HideCapturedPiecesInBoardMenuItem.state = NSControl.StateValue(rawValue:
<                                          BoolToInt(!HideCapturedPiecesInBoard))
---
>       // update the main window //
>       UpdateMainWindow()
5094a5192,5208
>    // hide the captured pieces in the panel //
>    func HideCapturedPiecesInPanelAction()
>    {
>       // update the captured pieces white view hidden state //
>       CapturedPiecesWhiteView.isHidden = HideCapturedPiecesInPanel
>
>       // update the captured pieces black view hidden state //
>       CapturedPiecesBlackView.isHidden = HideCapturedPiecesInPanel
>
>       // update the hide captured pieces in panel preference //
>       AppPreferences.set(HideCapturedPiecesInPanel, forKey:"HideCapturedPiecesInPanel")
>
>       // update the hide captured pieces in panel menu item state //
>       HideCapturedPiecesInPanelMenuItem.state = NSControl.StateValue(rawValue:
>                                          BoolToInt(!HideCapturedPiecesInPanel))
>    }
>
5227a5342,5344
>       // get the current hide pieces in panel value //
>       HidePieces = CGFloat(BoolToInt(HideCapturedPiecesInPanel))
>
5232,5233c5349,5362
<          GameClock.frame = NSMakeRect(717, 527, 224, 104)
<
---
>
>          GameClock.frame = NSMakeRect(674, 525, 310, 130)
>
>          // set the captured white pieces size //
>          CapturedPiecesWhiteView.Width = 20
>
>          // update the captured pieces white view frame //
>          CapturedPiecesWhiteView.frame = NSMakeRect(692, 500 + HideClock * 120, 308, 20)
>
>          // set the captured black pieces size //
>          CapturedPiecesBlackView.Width = 20
>
>          // update the captured pieces black view frame //
>          CapturedPiecesBlackView.frame = NSMakeRect(692, 480 + HideClock * 120, 308, 20)
5235,5236c5364,5365
<          PGNMovesTableView.enclosingScrollView?.frame = NSMakeRect(692, 30, 275, 235 + HideClock *
<                                                                    60)
---
>          PGNMovesTableView.enclosingScrollView?.frame = NSMakeRect(692, 30, 275, 215 + HideClock *
>                                                                    60 + HidePieces * 20)
5239,5240c5368,5370
<          GameInfoTableView.enclosingScrollView?.frame = NSMakeRect(692, 275 + HideClock * 60, 275,
<                                                                    235 + HideClock * 60)
---
>          GameInfoTableView.enclosingScrollView?.frame = NSMakeRect(692, 255 + HideClock * 60 +
>                                                                    HidePieces * 20, 275, 215 +
>                                                                    HideClock * 60 + HidePieces * 20)
5246,5247c5376,5389
<          GameClock.frame = NSMakeRect(980, 527, 224, 104)
<
---
>
>          GameClock.frame = NSMakeRect(795, 525, 310, 130)
>
>          // set the captured white pieces size //
>          CapturedPiecesWhiteView.Width = 35
>
>          // update the captured pieces white view frame //
>          CapturedPiecesWhiteView.frame = NSMakeRect(670, 485 + HideClock * 120, 590, 35)
>
>          // set the captured black pieces size //
>          CapturedPiecesBlackView.Width = 35
>
>          // update the captured pieces black view frame //
>          CapturedPiecesBlackView.frame = NSMakeRect(670, 450 + HideClock * 120, 590, 35)
5249c5391,5392
<          PGNMovesTableView.enclosingScrollView?.frame = NSMakeRect(670, 30, 275, 600)
---
>          PGNMovesTableView.enclosingScrollView?.frame = NSMakeRect(670, 30, 275, 480 + HideClock *
>                                                                    120 + HidePieces * 70)
5252,5253c5395,5396
<          GameInfoTableView.enclosingScrollView?.frame = NSMakeRect(955, 30, 275, 480 + HideClock *
<                                                                    120)
---
>          GameInfoTableView.enclosingScrollView?.frame = NSMakeRect(955, 30, 275, 480 + HideClock *
>                                                                    120 + HidePieces * 70)
5408c5551
<             HideCapturedPiecesInBoardMenuItem || MenuItem == HideIllustrationsMenuItem ||
---
>             HideCapturedPiecesInPanelMenuItem || MenuItem == HideIllustrationsMenuItem ||
6003,6004c6146,6147
<       // if scroll to visible count is greater than four //
<       if ScrollToVisibleCount > 4
---
>       // if scroll to visible count is greater than eight //
>       if ScrollToVisibleCount > 8
6028,6029c6171,6172
<       // if the alpha value of the game clock is greater than 0.25 //
<       if GameClock.alphaValue > 0.25
---
>       // if the alpha value of the game clock is greater than zero //
>       if GameClock.alphaValue > 0
6053c6196
<             ElapsedWhiteTime += 2
---
>             ElapsedWhiteTime += 1
6059c6202
<             ElapsedBlackTime += 2
---
>             ElapsedBlackTime += 1
7143,7144c7286,7287
<       // update the hide pieces in board value //
<       HideCapturedPiecesInBoard = PreferenceBool("HideCapturedPiecesInBoard", Default:false)
---
>       // update the hide pieces in panel value //
>       HideCapturedPiecesInPanel = PreferenceBool("HideCapturedPiecesInPanel", Default:false)
7146,7148c7289,7290
<       // update the hide captured pieces in board menu item state //
<       HideCapturedPiecesInBoardMenuItem.state = NSControl.StateValue(rawValue:
<                                          BoolToInt(!HideCapturedPiecesInBoard))
---
>       // update the captured pieces in panel hidden state //
>       HideCapturedPiecesInPanelAction()
8424,8437d8565
<       // allocate the movie info layer frame //
<       MovieInfoLayer = CGLayerOfSize(CGBoardSize[2] * Scale)
<
<       // loop over thirty rectangles for the captured pieces //
<       for Count in 0 ..< 30
<       {
<          // append a captured piece rectangle //
<          CRects.append(CGRect(x:Scale * (30 + CGFloat(Count) * 20), y:Scale * 634, width:Scale *
<                   20, height:Scale * 20))
<       }
<
<       // set the rectangle for the captured pieces background //
<       CRectBackground = CGRect(x:0, y:Scale * 634, width:Scale * 660, height:Scale * 26)
<
8442c8570
<          CGBoardOffset[Count] = 30.0 / CGBoardSize[2] * CGBoardSize[Count]
---
>          CGBoardOffset[Count] = 30.0 / 660.0 * CGBoardSize[Count]
8481c8609
<       AppNameView.stringValue = "GreenChess 8.1.0"
---
>       AppNameView.stringValue = "GreenChess 8.0.8"
8698c8826,8832
<
---
>
>       // save the chess piece images to the captured white pieces view //
>       CapturedPiecesWhiteView.SetChessPieceSetImages(ChessPieceImages)
>
>       // save the chess piece images to the captured black pieces view //
>       CapturedPiecesBlackView.SetChessPieceSetImages(ChessPieceImages)
>
8824,8895d8957
<    // create the movie info layer frame //
<    func CreateMovieInfoLayer()
<    {
<       // set the current graphics context to be the movie info layer context //
<       NSGraphicsContext.current = NSGraphicsContext(cgContext:MovieInfoLayer!.context!, flipped:
<                                  false)
<
<       // set the current graphics context alpha to one //
<       NSGraphicsContext.current?.cgContext.setAlpha(1)
<
<       // draw the board with pieces layer into the current context //
<       NSGraphicsContext.current?.cgContext.draw(CGLayersBoardWithPieces[2], in:ViewBounds)
<
<       // set the movie info attributes //
<       MovieInfoAttributes = [NSAttributedStringKey.font:NSFont.boldSystemFont(ofSize:Scale * 18),
<                       NSAttributedStringKey.foregroundColor:NSColor.black]
<
<       // set the initial movie info tag position //
<       MovieInfoTagPosition = 583
<
<       // write the event tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.EventTitle, Label:"Event:  ")
<
<       // write the site tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.SiteLocation, Label:"Site:  ")
<
<       // write the date tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.DateGamePlayed, Label:"Date:  ")
<
<       // write the round tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.RoundNumber, Label:"Round:  ")
<
<       // write the white tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.WhiteName, Label:"White:  ")
<
<       // write the black tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.BlackName, Label:"Black:  ")
<
<       // write the result tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.EndResult, Label:"Result:  ")
<
<       // write the time control tag to the movie info frame if it exists //
<       WriteMovieInfoTag(Attributes.TimeControl, Label:"Time Control:  ")
<    }
<
<    // write the movie info tag to the movie info frame //
<    func WriteMovieInfoTag(_ PGNTag:String, Label:String)
<    {
<       // if the pgn tag is not empty //
<       if !PGNTag.isEmpty
<       {
<          // add the label to the attributed string //
<          AttributedMovieTag = NSAttributedString(string:Label + PGNTag, attributes:
<                               MovieInfoAttributes as? [NSAttributedStringKey:
<                               Any])
<
<          // set the text background color to white //
<          NSColor.white.setFill()
<
<          // draw the text background //
<          NSBezierPath(roundedRect:(NSMakeRect(Scale * 35, Scale * (MovieInfoTagPosition - 3),
<                AttributedMovieTag.size().width + Scale * 20, Scale * 26)), xRadius:Scale *
<                13, yRadius:Scale * 13).fill()
<
<          // draw the attributed string for the given tag //
<          AttributedMovieTag.draw(at:NSPoint(x:Scale * 45, y:Scale * MovieInfoTagPosition))
<
<          // decrement the movie info tag position //
<          MovieInfoTagPosition -= 75
<       }
<    }
9162c9224
<       // if chose to highlight captures //
---
>       // if chose to highlight captured pieces //
9244,9303d9305
<       // if set is two and not hiding captured pieces in board and not in position editor //
<       if Set == 2 && !HideCapturedPiecesInBoard && !InPositionEditor
<       {
<          // initialize the captured pieces count //
<          CapturedPieceCount = 0
<
<          // set the capture pieces background color to match that of the main window //
<          CGLayersBoardWithPieces[Set].context?.setFillColor(NSColor(patternImage:
<                                                 WindowBackgroundPatternScaled).cgColor)
<
<          // compute the total number of captured pieces //
<          CapturedPieceCount = CapturedWhitePieces.reduce(0, +) + CapturedBlackPieces.reduce(0, +)
<
<          // if have captured pieces //
<          if CapturedPieceCount > 0
<          {
<             // fill the background color for the captured pieces //
<             CGLayersBoardWithPieces[Set].context?.fill(CRectBackground)
<          }
<
<          // reset the captured pieces count back to zero //
<          CapturedPieceCount = 0
<
<          // loop over the list of captured white pieces to display //
<          for Index in 0 ..< 5
<          {
<             // loop over the number of piece captures for this particular piece type //
<             for _ in 0 ..< CapturedWhitePieces[Index]
<             {
<                // set the piece index for the captured piece //
<                ChessPieceImageIndex = 4 - Index + ChessPieceSetType * 12
<
<                // draw the captured piece in the view //
<                CGLayersBoardWithPieces[Set].context?.draw(ChessPiecesCG[ChessPieceImageIndex], in:
<                                     CRects[CapturedPieceCount])
<
<                // increment the captured piece count //
<                CapturedPieceCount += 1
<             }
<          }
<
<          // loop over the list of captured black pieces to display //
<          for Index in 0 ..< 5
<          {
<             // loop over the number of piece captures for this particular piece type //
<             for _ in 0 ..< CapturedBlackPieces[Index]
<             {
<                // set the piece index for the captured piece //
<                ChessPieceImageIndex = 10 - Index + ChessPieceSetType * 12
<
<                // draw the captured piece in the view //
<                CGLayersBoardWithPieces[Set].context?.draw(ChessPiecesCG[ChessPieceImageIndex], in:
<                                     CRects[CapturedPieceCount])
<
<                // increment the captured piece count //
<                CapturedPieceCount += 1
<             }
<          }
<       }
9313c9315
<    // get a cgimage from a board cg layer //
---
>    // get an nsimage from a board cg layer //
9323,9332d9324
<    // get a cgimage from a movie info cg layer //
<    func CGImageFromMovieInfoLayer() -> CGImage
<    {
<       // draw the movie info layer into the context //
<       CGContexts[2].draw(MovieInfoLayer, in:ViewBounds)
<
<       // return a cg image from the context //
<       return CGContexts[2].makeImage()!
<    }
<
10692,10702d10683
<    // write a movie info frame //
<    func WriteMovieInfoFrame()
<    {
<       // wait until the adaptor is ready for more media to be added //
<       while !(Adaptor?.assetWriterInput.isReadyForMoreMediaData)! {}
<
<       // append the pixel buffer with duration of five seconds //
<       Adaptor!.append(PixelBufferFromCGImage(CGImageFromMovieInfoLayer()), withPresentationTime:
<                    CMTimeMake(0, 1))
<    }
<
10711c10692
<                    CMTimeMake(Int64(3 + FrameNumber * 2), 1))
---
>                    CMTimeMake(Int64(FrameNumber * 2), 1))
10717c10698
<    // prepare to write a movie of the given size //
---
>    // prepare to write a movie of the give size //
10719c10700
<
---
>
11581a11563,11568
>
>       // hide the captured pieces white view //
>       CapturedPiecesWhiteView.alphaValue = 0
>
>       // hide the captured pieces black view //
>       CapturedPiecesBlackView.alphaValue = 0
11588,11590d11574
<
<       // hide the game clock //
<       GameClock.alphaValue = 0
11722a11707,11712
>
>       // unhide the captured pieces white view //
>       CapturedPiecesWhiteView.alphaValue = 1
>
>       // unhide the captured pieces black view //
>       CapturedPiecesBlackView.alphaValue = 1
12326a12317
>
13429c13420
<       GameClockPreview.frame = NSMakeRect(218, 275, 224, 104)
---
>       GameClockPreview.frame = NSMakeRect(175, 230, 310, 130)
17354,17366d17344
<       // if recording a movie //
<       if RecordingMovie
<       {
<          // replay all the moves to get to the last frame //
<          ReplayMoves(0, End:MoveList.count, Display:false)
<
<          // create the movie info layer //
<          CreateMovieInfoLayer()
<
<          // write the movie info frame //
<          WriteMovieInfoFrame()
<       }
<
17374,17375c17352,17353
<       // add the initial position frame //
<       AddFrame(BoolToInt(RecordingMovie))
---
>       // add a frame //
>       AddFrame(0)
17412c17390
<             AddFrame(Index + 1 + BoolToInt(RecordingMovie))
---
>             AddFrame(Index + 1)
17514,17519c17492,17493
<       // if not recording a movie //
<       if !RecordingMovie
<       {
<          // print the game info //
<          PrintGameInfo()
<       }
---
>       // print the game info //
>       PrintGameInfo()
17562a17537,17548
>
>       // if not in game gallery, not recording movie, and not annotating //
>       if !InGameGallery && !RecordingMovie && !AnnotatingGame
>       {
>          // update the captured white pieces display //
>          CapturedPiecesWhiteView.UpdateCapturedPiecesDisplay(CapturedWhitePieces, White:true, Set:
>                                                   ChessPieceSetType)
>
>          // update the captured black pieces display //
>          CapturedPiecesBlackView.UpdateCapturedPiecesDisplay(CapturedBlackPieces, White:false, Set:
>                                                   ChessPieceSetType)
>       }
```