```
2c2
< // Swift source code of main.swift Version 8.1.5 at greenchess.com //
---
> // Swift source code of main.swift Version 8.1.4 at greenchess.com //
641,656c641,644
<     // the current diagram position //
<     var CurrentDiagramPosition:Int = 0
<
<     // a list of diagram paths for drawing //
<     var DiagramPaths:[CGMutablePath] = []
<
<     // a list of diagram path colors //
<     var DiagramPathColors:[CGColor] = []
<
<     // the diagram path location //
<     var PathLocation:NSPoint = CGPoint()
<
<     // the chess board //
<     @IBOutlet var Board:GreenChess!
<
<     // the core graphics context //
---
>     // the window resolution scale //
>     var Scale:CGFloat = 2
>
>     // the cg context //
659,661d646
<     // the snapshot image representation //
<     var SnapshotImageRep:NSBitmapImageRep!
<
665,666c650,651
<     // set that not initially a valid path //
<     var ValidPath:Bool = false
---
>     // set that not initially a valid shape //
>     var ValidShape:Bool = false
670a656,658
>     // set that haven't initially entered tracking rectangle //
>     var Entered:Bool = false
>
674,675c662,663
<     // the current cursor location //
<     var Cursor:NSPoint = NSPoint()
---
>     // the shape layers //
>     var Shapes:[CAShapeLayer] = []
677,678c665,666
<     // the previous location //
<     var PreviousLocation:NSPoint = NSPoint()
---
>     // a shape layer //
>     var Shape:CAShapeLayer = CAShapeLayer()
680,681c668,669
<     // the rectangles to draw the boards in //
<     var Rects:[CGRect] = []
---
>     // a shape path //
>     var ShapePath:CGMutablePath = CGMutablePath()
683,690c671,672
<     // the core graphics view bounds //
<     var ViewBounds:CGRect = CGRect(x:0, y:0, width:1, height:1)
<
<     // the shifted core graphics view bounds //
<     var ShiftedViewBounds:CGRect = CGRect(x:0, y:0, width:1, height:1)
<
<     // the board bounds //
<     var BoardBounds:CGRect = CGRect(x:0, y:0, width:1, height:1)
---
>     // the animation duration //
>     var Duration:CGFloat = 1.0
692,696c674,678
<     // the window resolution scale //
<     var Scale:CGFloat = 2
<
<     // the board size //
<     var BoardSize:CGFloat = 0
---
>     // the snapshot bitmap image representation //
>     var SnapshotBitmap:NSBitmapImageRep!
>
>     // the starting x value //
>     var StartX:CGFloat = 0
698,699c680,681
<     // the frame size //
<     var FrameSize:CGFloat = 0
---
>     // whether going right or not //
>     var GoingRight:Bool = false
701,702c683,684
<     // the gallery frames //
<     var Frames:[CGLayer] = []
---
>     // the animation start time //
>     var StartTime:Date = Date()
704,705c686,687
<     // the gallery layer //
<     var GalleryLayer:CGLayer!
---
>     // the elapsed time of the animation //
>     var ElapsedTime:CGFloat = 0
707,708c689,690
<     // the diagram layer for drawing //
<     var DiagramLayer:CGLayer!
---
>     // whether animating or not //
>     var Animating:Bool = false
709a692,694
>     // whether reviewing or not //
>     var Reviewing:Bool = false
>
715,720d699
<
<     // a list index //
<     var Index:Int = 0
<
<     // the number of frames that can span the width of the view //
<     var FramesWide:Int = 0
722,723c701,702
<     // the current position in the gallery //
<     var CurrentPosition:Int = 0
---
>     // create an animation layer //
>     var AnimationLayer:CALayer = CALayer()
725,726c704,705
<     // the maximum position possible in the gallery //
<     var MaxPosition:Int = 0
---
>     // create an animation //
>     var Animation:CABasicAnimation = CABasicAnimation(keyPath:"position")
728,729c707,708
<     // the maximum number of frames in the gallery //
<     var MaxFrames:Int = 0
```

```
---
>     // the x location value //
>     var X:CGFloat = 0
731,732c710,714
<     // whether is a wraparound gallery or not //
<     var Wraparound:Bool = false
---
>     // the y location value //
>     var Y:CGFloat = 0
> 
>     // the x location center of the animation layer //
>     var CenterX:CGFloat = 0
734,735c716,717
<     // whether reviewing gallery or not //
<     var Reviewing:Bool = false
---
>     // the maximum animation layer offset //
>     var MaxOffset:CGFloat = 0
737,738c719,720
<     // whether recording movie //
<     var RecordingMovie:Bool = false
---
>     // the image width //
>     var ImageWidth:CGFloat = 0
740,741c722,723
<     // the diagram cursor //
<     var DiagramCursor:NSCursor = NSCursor()
---
>     // the width of one frame in the animation //
>     var FrameWidth:CGFloat = 0
743,744c725,726
<     // the gallery diagram color //
<     var GalleryColor:NSColor = NSColor()
---
>     // the current image //
>     var Image:CGImage!
746,757c728,729
<     // clear the diagram the user drew on //
<     func ClearDiagram()
<     {
<        // remove all the diagram paths //
<        DiagramPaths.removeAll()
< 
<        // remove all the diagram path colors //
<        DiagramPathColors.removeAll()
< 
<        // refresh the diagram layer //
<        RefreshDiagramLayer()
<     }
---
>     // whether is a wraparound or not //
>     var Wraparound:Bool = false
759,774c731,735
<     // undo the diagram path //
<     func UndoPath()
<     {
<        // if there is a diagram path to undo //
<        if !DiagramPaths.isEmpty
<        {
<           // remove the last diagram path //
<           DiagramPaths.removeLast()
< 
<           // remove the last diagram color //
<           DiagramPathColors.removeLast()
<        }
< 
<        // refresh the diagram layer //
<        RefreshDiagramLayer()
<     }
---
>     // the diagram cursor //
>     var DiagramCursor:NSCursor = NSCursor()
> 
>     // the chess board //
>     @IBOutlet var Board:GreenChess!
776,777c737,738
<     // update the gallery color //
<     func UpdateGalleryColor(_ Color:String)
---
>     // set the diagram cursor //
>     func SetDiagramCursor(_ MarkerColor:String)
779,781d739
<        // play the game sound of a click //
<        Board.PlayGameSoundThread("click")
< 
786c744
<        DiagramCursor = NSCursor.init(image:NSImage(named:NSImage.Name("marker_\(Color)_icon"))!,
---
>        DiagramCursor = NSCursor.init(image:NSImage(named:NSImage.Name("marker_\(MarkerColor)_icon"))!,
788,836d745
< 
<        // if the color is cyan //
<        if Color == "cyan"
<        {
<           // set the gallery color to cyan //
<           GalleryColor = NSColor.cyan
<        }
<        // if the color is blue //
<        else if Color == "blue"
<        {
<           // set the gallery color to blue //
<           GalleryColor = NSColor.blue
<        }
<        // if the color is yellow //
<        else if Color == "yellow"
<        {
<           // set the gallery color to yellow //
<           GalleryColor = NSColor.yellow
<        }
<        // if the color is black //
<        else if Color == "black"
<        {
<           // set the gallery color to black //
<           GalleryColor = NSColor.black
<        }
<        // if the color is white //
<        else if Color == "white"
```

```
<       {
<          // set the gallery color to white //
<          GalleryColor = NSColor.white
<       }
<       // if the color is magenta //
<       else if Color == "magenta"
<       {
<          // set the gallery color to magenta //
<          GalleryColor = NSColor.magenta
<       }
<       // if the color is green //
<       else if Color == "green"
<       {
<          // set the gallery color to green //
<          GalleryColor = NSColor.green
<       }
<       // if the color is red //
<       else if Color == "red"
<       {
<          // set the gallery color to red //
<          GalleryColor = NSColor.red
<       }
839,840c748,749
<    // the draw rect function //
<    override func draw(_ Rect:NSRect)
---
>    // set the current animation rate //
>    func SetAnimationRate(_ AnimationRate:Int)
842,850c751,752
<       // draw the galley layer into the current view context //
<       NSGraphicsContext.current?.cgContext.draw(GalleryLayer, in:bounds)
<
<       // if diagramming //
<       if Diagramming
<       {
<          // draw the diagram layer into the current view context //
<          NSGraphicsContext.current?.cgContext.draw(DiagramLayer, in:bounds)
<       }
---
>       // compute the duration for each frame to animate for //
>       Duration = (101.0 - CGFloat(AnimationRate)) / 10.0
853,854c755,756
<    // get the current location //
<    func GetLocation(_ Event:NSEvent)
---
>    // take a snapshot of the view //
>    func TakeSnapshot()
856,880c758,759
<       // get the location in the window //
<       Location = Event.locationInWindow
<
<       // save the cursor location //
<       Cursor = Location
<
<       // decrement the vertical offset of the gallery view due to the gallery slider position //
<       Location.y -= 25
<
<       // update the cursor //
<       UpdateCursor()
<
<       // keep the marker in bounds //
<       KeepMarkerInBounds()
<
<       // set the path location //
<       PathLocation = CGPoint(x:(Location.x + FrameSize * CGFloat(CurrentPosition)) * Scale,
<                             y:Location.y * Scale)
<    }
<
<    // if the mouse entered the tracking rectangle //
<    override func mouseEntered(with Event:NSEvent)
<    {
<       // update the cursor //
<       Cursor = Event.locationInWindow
---
>       // set the snapshot bitmap //
>       SnapshotBitmap = bitmapImageRepForCachingDisplay(in:bounds)!
882,883c761,762
<       // update the cursor //
<       UpdateCursor()
---
>       // draw the specified view into the bitmap representation //
>       cacheDisplay(in:bounds, to:SnapshotBitmap)
886,887c765,766
<    // if the mouse exited the tracking rectangle //
<    override func mouseExited(with Event:NSEvent)
---
>    // animation did start delegate //
>    func animationDidStart(_ TheAnimation:CAAnimation)
889,890c768,772
<       // update the cursor //
<       Cursor = Event.locationInWindow
---
>       // set the arrow cursor //
>       NSCursor.arrow.set()
>
>       // set that not currently a valid shape //
>       ValidShape = false
892,893c774,775
<       // update the cursor //
<       UpdateCursor()
---
>       // get the animation start time //
>       StartTime = Date()
896,897c778,779
<    // refresh the diagram layer //
<    func RefreshDiagramLayer()
---
>    // animation did stop delegate //
>    func animationDidStop(_ TheAnimation:CAAnimation, finished Flag:Bool)
899,911c781,782
<       // translate the diagram layer context to the current diagram position //
<       DiagramLayer.context?.translateBy(x:CGFloat(CurrentDiagramPosition - CurrentPosition) *
<                                       FrameSize * Scale, y:0)
<
<       // set the current diagram position to the current position //
<       CurrentDiagramPosition = CurrentPosition
<
<       // clear the diagram layer context //
<       DiagramLayer.context?.clear(CGRect(x:FrameSize * Scale * CGFloat(CurrentPosition), y:0, width:
<                                  Width * Scale, height:Height * Scale))
<
<       // loop over all the diagram paths //
<       for Index in 0 ..< DiagramPaths.count
---
>       // if in tracking rectangle and diagramming //
>       if Entered && Diagramming
913,917c784,792
<          // begin a new diagram path //
<          DiagramLayer.context?.beginPath()
<
<          // set the stroke color for this path //
<          DiagramLayer.context?.setStrokeColor(DiagramPathColors[Index])
```

```
---
>          // set the diagram cursor //
>          DiagramCursor.set()
>       }
>
>       // set that no longer animating //
>       Animating = false
>
>       // update the gallery slider value //
>       Board.GameGallerySlider.doubleValue = SliderValue()
919,923c794,801
<          // add the path to the diagram layer context //
<          DiagramLayer.context?.addPath(DiagramPaths[Index])
<
<          // stroke the current path //
<          DiagramLayer.context?.strokePath()
---
>       // if x went past the minimum value or not reviewing //
>       if (X <= -MaxOffset + CenterX) || !Reviewing
>       {
>          // set that no longer reviewing //
>          Reviewing = false
>
>          // set the gallery buttons to true //
>          Board.SetGalleryButtonsToState(true)
925,927c803,804
<
<       // if not recording a movie //
<       if !RecordingMovie
---
>       // if reviewing //
>       else
929,930c806,807
<          // update the display in the bounds //
<          displayIgnoringOpacity(bounds)
---
>          // go right in the game gallery //
>          Board.GameGalleryRightAction(self)
934,935c811,812
<    // override mouse down //
<    override func mouseDown(with Event:NSEvent)
---
>    // stop the animation //
>    func StopAnimation()
937,938c814,815
<       // if reviewing or not diagramming //
<       if Reviewing || !Diagramming
---
>       // if not animating //
>       if !Animating
940,943c817
<          // set that not a valid path //
<          ValidPath = false
<
<          // exit function since reviewing or not diagramming //
---
>          // exit function since not animating //
946,948d819
<
<       // get the current location //
<       GetLocation(Event)
950,951c821,837
<       // set that the current path is valid //
<       ValidPath = true
---
>       // get the elapsed time of the animation //
>       ElapsedTime = min(Duration, CGFloat(abs(StartTime.timeIntervalSinceNow)))
>
>       // if going right and the animation duration is greater than zero //
>       if GoingRight && (Animation.duration > 0)
>       {
>          // set the new x location value //
>          X += (CGFloat(Animation.duration) - ElapsedTime) / CGFloat(Animation.duration) *
>               min(FrameWidth, abs(StartX - X))
>       }
>       // if going left and animation duration is greater than zero //
>       else if !GoingRight && (Animation.duration > 0)
>       {
>          // set the new x location value //
>          X -= (CGFloat(Animation.duration) - ElapsedTime) / CGFloat(Animation.duration) *
>               min(FrameWidth, abs(StartX - X))
>       }
953,969c839,840
<       // play the sound of a marker pen starting to write //
<       Board.PlayGameSoundThread("marker")
<
<       // append a path to the diagram paths //
<       DiagramPaths.append(CGMutablePath())
<
<       // append a path color to the diagram path colors //
<       DiagramPathColors.append(GalleryColor.cgColor)
<
<       // set the diagram layer stroke color //
<       DiagramLayer.context?.setStrokeColor(GalleryColor.cgColor)
<
<       // move the diagram path to the first location //
<       DiagramPaths[DiagramPaths.count - 1].move(to:PathLocation)
<
<       // set the diagram path starting point in the context //
<       DiagramLayer.context?.move(to:PathLocation)
---
>       // remove the animation effects //
>       CATransaction.setDisableActions(true)
971,972c842,849
<       // save the location //
<       PreviousLocation = Location
---
>       // update the animation layer position //
>       AnimationLayer.position = CGPoint(x:X, y:Y)
>
>       // set that no longer reviewing //
>       Reviewing = false
>
>       // remove all the gallery view animations //
>       AnimationLayer.removeAllAnimations()
975,976c852,853
<    // override mouse dragged //
<    override func mouseDragged(with Event:NSEvent)
---
>    // if the mouse entered the tracking rectangle //
>    override func mouseEntered(with Event:NSEvent)
978,982c855,859
<       // get the current location //
<       GetLocation(Event)
<
<       // if reviewing or not diagramming or not a valid path //
<       if Reviewing || !Diagramming || !ValidPath
---
```

```
>        // set that entered the tracking rectangle //
>        Entered = true
>
>        // if diagramming and not animating //
>        if Diagramming && !Animating
984,985c861,862
<           // return since reviewing or not diagramming or not a valid path //
<           return
---
>           // set the cursor to the diagram cursor //
>           DiagramCursor.set()
987,1000d863
<
<        // process the diagram path //
<        ProcessDiagramPath()
<
<        // display the area that was drawn in //
<        displayIgnoringOpacity(NSMakeRect(min(PreviousLocation.x - 4, Location.x - 4),
<                              min(PreviousLocation.y - 4, Location.y - 4), abs(PreviousLocation.x -
<                              Location.x) + 8, abs(PreviousLocation.y - Location.y) + 8))
<
<        // move the diagram layer path to the path location //
<        DiagramLayer.context?.move(to:PathLocation)
<
<        // save the location //
<        PreviousLocation = Location
1003,1004c866,867
<     // process the diagram path //
<     func ProcessDiagramPath()
---
>     // if the mouse exited the tracking rectangle //
>     override func mouseExited(with Event:NSEvent)
1006,1013c869,873
<        // add a line to the diagram path location //
<        DiagramPaths[DiagramPaths.count - 1].addLine(to:PathLocation)
<
<        // add a line to the diagram layer context path //
<        DiagramLayer.context?.addLine(to:PathLocation)
<
<        // stroke the diagram layer path //
<        DiagramLayer.context?.strokePath()
---
>        // set that no longer in the tracking rectangle //
>        Entered = false
>
>        // set the cursor back to the arrow cursor //
>        NSCursor.arrow.set()
1015,1017c875,877
<
<     // override mouse up //
<     override func mouseUp(with Event:NSEvent)
---
>
>     // get the current location //
>     func GetLocation(_ Event:NSEvent)
1019,1023c879,889
<        // get the current location //
<        GetLocation(Event)
<
<        // if reviewing or not diagramming or not a valid path //
<        if Reviewing || !Diagramming || !ValidPath
---
>        // get the location in the window //
>        Location = Event.locationInWindow
>
>        // adjust the location x value //
>        Location.x += CenterX - X - frame.origin.x
>
>        // adjust the location y value //
>        Location.y -= frame.origin.y
>
>        // if the location x value is too low //
>        if Location.x <= CenterX - X
1025,1026c891,892
<           // return since not reviewing or not diagramming or not a valid path //
<           return
---
>           // reset the location x value //
>           Location.x = CenterX - X + 3.0
1028,1046c894,896
<
<        // process the diagram path //
<        ProcessDiagramPath()
<
<        // set that no longer a valid path //
<        ValidPath = false
<
<        // refresh the diagram layer //
<        RefreshDiagramLayer()
<     }
<
<     // take a snapshot of the gallery //
<     func GallerySnapshot() -> NSBitmapImageRep
<     {
<        // draw the gallery layer into the context //
<        Context.draw(GalleryLayer, in:ViewBounds)
<
<        // if diagramming //
<        if Diagramming
---
>
>        // if location y value is too low //
>        if Location.y <= 0
1048,1049c898,899
<           // draw the diagram layer into the context //
<           Context.draw(DiagramLayer, in:ViewBounds)
---
>           // reset the location y value //
>           Location.y = 3.0
1051,1066c901,903
<
<        // get the snapshot image representation from the context //
<        SnapshotImageRep = NSBitmapImageRep(cgImage:Context.makeImage()!)
<
<        // set the snapshot image representation size //
<        SnapshotImageRep.size = bounds.size
<
<        // return the bitmap image rep //
<        return SnapshotImageRep
<     }
<
<     // convert a boolean to an integer //
<     func BoolToInt(_ BoolVar:Bool) -> Int
<     {
<        // if boolean is true //
<        if BoolVar
---
```

```
>
>       // if the location x value is too high //
>       if Location.x >= Width + CenterX - X
1068,1069c905,906
<          // return one //
<          return 1
---
>          // reset the location x value //
>          Location.x = Width + CenterX - X - 3.0
1071c908,923
<       // boolean is false //
---
>
>       // if the location y value is too high //
>       if Location.y >= Height
>       {
>          // reset the location y value //
>          Location.y = Height - 3.0
>       }
>
>       // if location is inside the tracking rectangle //
>       if Location.x > (CenterX - X + 3.0) && Location.x < (Width + CenterX - X - 3.0) && Location.y
>          > 3.0 && Location.y < (Height - 3.0)
>       {
>          // set the diagram cursor //
>          DiagramCursor.set()
>       }
>       // location is outside the tracking rectangle //
1074,1075c926,927
<          // return zero //
<          return 0
---
>          // set the arrow cursor //
>          NSCursor.arrow.set()
1079,1080c931,932
<    // create a cg layer of the given size //
<    func CGLayerOfSize(_ Width:CGFloat, Height:CGFloat) -> CGLayer
---
>    // undo a shape //
>    func UndoShape()
1082,1096c934,938
<       // return a cg layer of the given size /
<       return CGLayer(CGContext(data:nil, width:Int(Width), height:Int(Height), bitsPerComponent:8,
<                      bytesPerRow:0, space:CGColorSpaceCreateDeviceRGB(), bitmapInfo:CGImageAlphaInfo
<                      .noneSkipFirst.rawValue)!, size:CGSize(width:Width, height:Height), auxiliaryInfo:
<                      nil)!
<    }
<
<    // add a new frame //
<    func AddFrame(_ FrameNumber:Int, Layer:CGLayer)
<    {
<       // draw in a new frame //
<       Frames[FrameNumber].context?.draw(Layer, in:CGRect(x:0, y:0, width:BoardSize, height:BoardSize))
<
<       // the max number of frames //
<       MaxFrames = FrameNumber + 1
---
>       // if there is a shape to undo //
>       if Shapes.count > 0
>       {
>          // remove the shape layer from the super layer //
>          Shapes[Shapes.count - 1].removeFromSuperlayer()
1098,1099c940,942
<       // set the maximum gallery frame position //
<       MaxPosition = max(MaxFrames - FramesWide * (BoolToInt(Wraparound) + 1), 0)
---
>          // remove most current shape from the shape list //
>          Shapes.removeLast()
>       }
1102,1103c945,946
<    // go left in the game gallery //
<    func GoLeft()
---
>    // erase all the shapes //
>    func EraseShapes()
1105,1119c948,949
<       // invalidate the diagram path //
<       ValidPath = false
<
<       // if the current position is greater than zero //
<       if CurrentPosition > 0
<       {
<          // decrement the current position //
<          CurrentPosition -= 1
<
<          // update the gallery //
<          UpdateGallery()
<       }
<
<       // if the game gallery slider double value is other than zero //
<       if Board.GameGallerySlider.doubleValue != 0
---
>       // loop over all the shapes //
>       for Index in 0 ..< Shapes.count
1121,1122c951,952
<          // update the game gallery slider to the current position //
<          Board.GameGallerySlider.intValue = Int32(CurrentPosition)
---
>          // remove the shape from the super layer //
>          Shapes[Index].removeFromSuperlayer()
1123a954,956
>
>       // remove all the shapes from the shape list //
>       Shapes.removeAll()
1126,1127c959,960
<    // update the cursor //
<    func UpdateCursor()
---
>    // override mouse down //
>    override func mouseDown(with Event:NSEvent)
1129,1131c962,963
<       // if the cursor is outside the tracking rectangle or reviewing or not diagramming //
<       if Cursor.x < 0 || Cursor.x > Width || Cursor.y < 25 || Cursor.y > Height + 25 || Reviewing ||
<          !Diagramming
---
>       // if animating or not diagramming //
>       if Animating || !Diagramming
1133,1140c965,966
<          // set the arrow cursor //
<          NSCursor.arrow.set()
<       }
<       // inside the tracking rectangle and diagramming and not reviewing //
<       else
<       {
<          // set the diagram cursor //
<          DiagramCursor.set()
---
```

```
>           // exit function since animating or not diagramming //
>           return
1141a968,1009
>
>        // set that the current shape is valid //
>        ValidShape = true
>
>        // play the sound of a marker pen starting to write //
>        Board.PlayGameSoundThread("marker")
>
>        // get the location for the shape //
>        GetLocation(Event)
>
>        // create a shape layer //
>        Shape = CAShapeLayer()
>
>        // create a new mutable shape path //
>        ShapePath = CGMutablePath()
>
>        // set the shape line width //
>        Shape.lineWidth = 4
>
>        // set the shape line cap style to round //
>        Shape.lineCap = CAShapeLayerLineCap(rawValue:"kCALineCapRound")
>
>        // set the shape line join style to round //
>        Shape.lineJoin = CAShapeLayerLineJoin(rawValue:"kCALineJoinRound")
>
>        // set the shape stroke color //
>        Shape.strokeColor = Board.GalleryColor.cgColor
>
>        // set the shape fill color //
>        Shape.fillColor = NSColor.clear.cgColor
>
>        // set the shape path starting point //
>        ShapePath.move(to:CGPoint(x:Location.x, y:Location.y))
>
>        // assign the shape path to the shape //
>        Shape.path = ShapePath
>
>        // append a new shape layer //
>        Shapes.append(Shape)
>
>        // add the new shape layer to the animation layer //
>        AnimationLayer.addSublayer(Shape)
1144,1145c1012,1013
<     // go right in the game gallery //
<     func GoRight()
---
>     // override mouse dragged //
>     override func mouseDragged(with Event:NSEvent)
1147,1176c1015,1016
<        // invalidate the diagram path //
<        ValidPath = false
<
<        // if the current position is less than the maximum position //
<        if CurrentPosition < MaxPosition
<        {
<           // increment the current position by one //
<           CurrentPosition += 1
<
<           // if reached the maximum position and reviewing the gallery //
<           if CurrentPosition == MaxPosition && Reviewing
<           {
<              // stop the gallery review timer //
<              Board.GalleryReviewTimer.invalidate()
<
<              // set that no longer reviewing //
<              Reviewing = false
<
<              // set the gallery buttons to state true //
<              Board.SetGalleryButtonsToState(true)
<
<              // update the cursor //
<              UpdateCursor()
<           }
<
<           // update the gallery //
<           UpdateGallery()
<        }
<        // if already at maximum position and reviewing gallery //
<        else if Reviewing
---
>        // if animating or not diagramming or not a valid shape //
>        if Animating || !Diagramming || !ValidShape
1178,1182c1018,1019
<           // set the current position to zero //
<           CurrentPosition = 0
<
<           // update the gallery //
<           UpdateGallery()
---
>           // return since animating or not diagramming or not a valid shape //
>           return
1183a1021,1023
>
>        // get the location for the shape //
>        GetLocation(Event)
1185,1190c1025,1029
<        // if the game gallery slider double value is other than the maximum position or reviewing //
<        if Board.GameGallerySlider.doubleValue != Double(MaxPosition) || Reviewing
<        {
<           // update the game gallery slider to the current value //
<           Board.GameGallerySlider.intValue = Int32(CurrentPosition)
<        }
---
>        // add the line to the shape path //
>        ShapePath.addLine(to:CGPoint(x:Location.x, y:Location.y))
>
>        // assign the shape path to the shape //
>        Shapes[Shapes.count - 1].path = ShapePath
1193,1194c1032,1033
<     // create a movie frame //
<     func MovieFrame() -> CGImage
---
>     // override mouse up //
>     override func mouseUp(with Event:NSEvent)
1196,1197c1035,1036
<        // if not set to recording movie //
<        if !RecordingMovie
---
>        // if animating or not diagramming or not a valid shape //
>        if Animating || !Diagramming || !ValidShape
1199,1203c1038,1042
<           // set the current position to zero //
<           CurrentPosition = 0
<
<           // set that recording a movie //
<           RecordingMovie = true
---
```

```
>          // set that no longer a valid shape //
>          ValidShape = false
>
>          // return since not animating or not diagramming or not a valid shape //
>          return
1204a1044,1052
>
>       // get the location for the shape //
>       GetLocation(Event)
>
>       // add the line to the shape path //
>       ShapePath.addLine(to:CGPoint(x:Location.x, y:Location.y))
>
>       // assign the shape path to the shape //
>       Shapes[Shapes.count - 1].path = ShapePath
1206,1207c1054,1095
<       // if is the first position of zero //
<       if CurrentPosition == 0
---
>       // set that no longer a valid shape //
>       ValidShape = false
>    }
>
>    // create the gallery image //
>    func CreateGalleryImage(FrameCount:Int)
>    {
>       // set that not currently animating //
>       Animating = false
>
>       // set that not currently reviewing //
>       Reviewing = false
>
>       // set that not currently diagramming //
>       Diagramming = false
>
>       // set that currently not a valid shape //
>       ValidShape = false
>
>       // set the gallery buttons to true //
>       Board.SetGalleryButtonsToState(true)
>
>       // set that want to rasterize the animation layer //
>       AnimationLayer.shouldRasterize = true
>
>       // set the animation layer rasterization to the window scale //
>       AnimationLayer.rasterizationScale = Scale
>
>       // allow the animation layer to draw asynchronously //
>       AnimationLayer.drawsAsynchronously = true
>
>       // set the animation layer contents scale to the window scale //
>       AnimationLayer.contentsScale = Scale
>
>       // get the view height //
>       Height = frame.size.height
>
>       // get the view width //
>       Width = frame.size.width
>
>       // if for a wraparound //
>       if Wraparound
1209,1218c1097,1098
<          // loop over all the boards for the current gallery view //
<          for Index in 0 ..< FramesWide * (BoolToInt(Wraparound) + 1)
<          {
<             // if the current board position is less than the maximum value allowed //
<             if Index + CurrentPosition < MaxFrames
<              if Index + CurrentPosition < MaxFrames
<              {
<                 // draw the frame into the specified rectangle in the view //
<                 GalleryLayer.context?.draw(Frames[CurrentPosition + Index], in:Rects[Index])
<              }
<           }
---
>           // set the image width for a wraparound //
>           ImageWidth = max(FrameWidth * CGFloat(FrameCount) - Width, Width)
1220c1100
<        // if not the first position in the gallery //
---
>        // for a gallery //
1223,1236c1103,1104
<           // shift the gallery layer to the left by one frame //
<           GalleryLayer.context?.draw(GalleryLayer, in:ShiftedViewBounds)
<
<           // draw the rightmost frame //
<           GalleryLayer.context?.draw(Frames[CurrentPosition + FramesWide - 1], in:Rects[FramesWide -
<                                      1])
<
<           // if for a wraparound gallery //
<           if Wraparound
<           {
<              // draw the rightmost frame if have a wraparound gallery //
<              GalleryLayer.context?.draw(Frames[CurrentPosition + FramesWide * 2 - 1], in:
<                                         Rects[FramesWide * 2 - 1])
<           }
---
>           // set the image width for a gallery //
>           ImageWidth = max(FrameWidth * CGFloat(FrameCount), Width)
1239,1240c1107,1108
<        // draw the gallery layer into the context //
<        Context.draw(GalleryLayer, in:ViewBounds)
---
>        // get the x location center of the animation layer //
>        CenterX = ImageWidth / 2.0
1242,1243c1110,1111
<        // refresh the diagram layer //
<        RefreshDiagramLayer()
---
>        // set the maximum animation layer offset value //
>        MaxOffset = ImageWidth - Width
1245,1250c1113,1122
<        // if diagramming //
<        if Diagramming
<        {
<           // draw the diagram layer into the view bounds //
<           Context.draw(DiagramLayer, in:ViewBounds)
<        }
---
>        // set the x initial location //
>        X = CenterX
>
>        // set the y location //
>        Y = Height / 2.0
>
>        // set the context for the image //
>        Context = CGContext(data:nil, width:Int(ImageWidth * Scale), height:Int(Height * Scale),
>                            bitsPerComponent:8, bytesPerRow:0, space:CGColorSpaceCreateDeviceRGB(),
>                            bitmapInfo:CGImageAlphaInfo.noneSkipFirst.rawValue)!
1252,1253c1124,1125
<        // increment the current position //
<        CurrentPosition += 1
---
>        // set the animation layer bounds //
>        AnimationLayer.bounds = CGRect(x:0, y:0, width:ImageWidth, height:Height)
1255,1256c1127,1134
<        // return a context cg image //
<        return Context.makeImage()!
---
>        // set the animation delegate //
>        Animation.delegate = self
>
>        // add a tracking rectangle for the cursor //
>        TrackingRectangle = addTrackingRect(bounds, owner:self, userData:nil, assumeInside:false)
>
>        // disable the cursor rectangles //
>        window!.disableCursorRects()
1259,1260c1137,1138
<     // erase the gallery //
<     func EraseGallery()
---
>     // finish up the gallery image //
>     func FinishUpGalleryImage()
1262,1263c1140,1144
<        // set that no longer reviewing the gallery //
<        Reviewing = false
---
>        // get the image from the context //
>        Image = Context.makeImage()!
>
>        // remove the animation effects //
>        CATransaction.setDisableActions(true)
1265,1269c1146,1172
<        // remove the tracking rectangle //
<        removeTrackingRect(TrackingRectangle)
<
<        // loop over all the frames //
<        for Index in 0 ..< MaxFrames
---
>        // set the animation layer contents to the image //
>        AnimationLayer.contents = Image
>
>        // reset the context to empty //
>        Context = CGContext(data:nil, width:Int(Scale), height:Int(Scale), bitsPerComponent:8,
>                            bytesPerRow:0, space:CGColorSpaceCreateDeviceRGB(), bitmapInfo:
>                            CGImageAlphaInfo.noneSkipFirst.rawValue)!
>     }
>
>     // add a frame //
>     func AddFrame(_ FrameNumber:Int)
>     {
>        // if in the game wraparound //
>        if Wraparound
>        {
>           // draw the frame into the top row of the wraparound context //
>           Context.draw(Board.CGLayersBoardWithPieces[1], in:CGRect(x:FrameWidth * Scale *
>                       CGFloat(FrameNumber), y:FrameWidth * Scale, width:FrameWidth * Scale, height:
>                       FrameWidth * Scale))
>
>           // draw the frame into the bottom row of the wraparound context //
>           Context.draw(Board.CGLayersBoardWithPieces[1], in:CGRect(x:Scale * (FrameWidth *
>                       CGFloat(FrameNumber) - Width), y:0, width:FrameWidth * Scale, height:
>                       FrameWidth * Scale))
>        }
>        // in the game gallery //
>        else
1271,1272c1174,1177
<           // clear the frame context //
<           Frames[Index].context?.clear(BoardBounds)
```

```
---
>          // draw the frame into the gallery context //
>          Context.draw(Board.CGLayersBoardWithPieces[1], in:CGRect(x:FrameWidth * Scale *
>                      CGFloat(FrameNumber), y:0, width:FrameWidth * Scale, height:FrameWidth *
>                      Scale))
1274,1276d1178
<
<       // clear the diagram //
<       ClearDiagram()
1279,1280c1181,1182
<    // reset the gallery view //
<    func ResetGallery()
---
>    // get the current slider value //
>    func SliderValue() -> Double
1282,1283c1184,1198
<       // set the current position back to the original position //
<       CurrentPosition = Int(Board.GameGallerySlider.intValue)
---
>       // return the current slider value //
>       return Double((CenterX - X) / MaxOffset)
>    }
>
>    // set the animation layer position //
>    func SetAnimationLayerPosition()
>    {
>       // remove all the animations from the animation layer //
>       AnimationLayer.removeAllAnimations()
>
>       // remove the animation effects //
>       CATransaction.setDisableActions(true)
>
>       // set the animation ending point //
>       Animation.toValue = NSValue(point:NSPoint(x:X, y:Y))
1285,1286c1200,1204
<       // update the gallery //
<       UpdateGallery()
---
>       // remove the animation effects //
>       CATransaction.setDisableActions(true)
>
>       // update the animation layer position //
>       AnimationLayer.position = CGPoint(x:X, y:Y)
1288,1289c1206,1210
<       // set recording movie to false //
<       RecordingMovie = false
---
>       // remove the animation effects //
>       CATransaction.setDisableActions(true)
>
>       // set the animation duration //
>       Animation.duration = Double(min(abs(StartX - X) / FrameWidth * Duration, Duration))
1292,1293c1213,1214
<    // update the gallery //
<    func UpdateGallery()
---
>    // update the game gallery view //
>    func UpdateGalleryView(_ SliderValue:CGFloat)
1295,1313c1216,1220
<       // clear the gallery context //
<       GalleryLayer.context?.clear(ViewBounds)
<
<       // loop over all the boards for the current gallery view //
<       for Index in 0 ..< FramesWide * (BoolToInt(Wraparound) + 1)
<       {
<          // if the current board position is less than the maximum value allowed //
<          if Index + CurrentPosition < MaxFrames
<          {
<             // draw the frame into the specified rectangle in the view //
<             GalleryLayer.context?.draw(Frames[CurrentPosition + Index], in:Rects[Index])
<          }
<       }
<
<       // refresh the diagram layer //
<       RefreshDiagramLayer()
<
<       // update the view display //
<       displayIgnoringOpacity(bounds)
---
>       // compute the x location value of the animation layer from the slider value //
>       X = CenterX - SliderValue * MaxOffset
>
>       // set the animation layer position //
>       SetAnimationLayerPosition()
1316,1317c1223,1224
<    // keep the marker pen in bounds //
<    func KeepMarkerInBounds()
---
>    // animate moving a frame to the right or the left //
>    func Animate(_ Right:Bool)
1319,1324c1226,1227
<       // if the x coordinate is less than 1 //
<       if Location.x < 1
<       {
<          // reset the location x coordinate to 1 //
<          Location.x = 1
<       }
---
>       // set if going right or not //
>       GoingRight = Right
1326,1327c1229,1231
<       // if the y coordinate is less than 1 //
<       if Location.y < 1
---
>       // if have reached the maximum or the minimum value of x and not reviewing //
>       if (((X <= -MaxOffset + CenterX && GoingRight) || (X >= CenterX && !GoingRight)) &&
>          !Reviewing) || (ImageWidth == Width)
1329,1330c1233,1234
<          // reset the location y coordinate to 1 //
<          Location.y = 1
---
>          // exit function //
>          return
1333,1338c1237,1241
<       // if the x coordinate is greater than width - 1 //
<       if Location.x > Width - 1
<       {
<          // reset the location x coordinate to width - 1 //
<          Location.x = Width - 1
<       }
---
>       // set that currently animating //
>       Animating = true
>
>       // update the gallery buttons //
>       Board.SetGalleryButtonsToState(!Reviewing)
1340,1341c1243,1244
<       // if the y coordinate is greater than height - 1 //
<       if Location.y > Height - 1
---
```

```
>       // if have reached the minimum value of x and reviewing //
>       if (X <= -MaxOffset + CenterX) && GoingRight && Reviewing
1343,1344c1246,1250
<          // reset the location y coordinate to height - 1 //
<          Location.y = Height - 1
---
>          // set x to the initial value //
>          X = CenterX
>
>          // update the gallery slider value //
>          Board.GameGallerySlider.doubleValue = SliderValue()
1346,1379d1251
<    }
<
<    // initialize the gallery //
<    func Initialize(_ Size:CGFloat)
<    {
<       // set the board size //
<       BoardSize = Size * Scale
<
<       // set the valid path to false //
<       ValidPath = false
<
<       // set the current diagram position to zero //
<       CurrentDiagramPosition = 0
<
<       // set that not currently reviewing //
<       Reviewing = false
<
<       // set that not currently diagramming //
<       Diagramming = false
<
<       // set the current position to zero //
<       CurrentPosition = 0
<
<       // set the view height //
<       Height = frame.size.height
<
<       // set the view width //
<       Width = frame.size.width
<
<       // initialize the diagram cursor //
<       DiagramCursor = NSCursor.arrow
<
<       // set the core graphics view bounds //
<       ViewBounds = CGRect(x:0, y:0, width:Width * Scale, height:Height * Scale)
1381,1383c1253,1254
<       // the shifted view bounds //
<       ShiftedViewBounds = CGRect(x:-Scale * FrameSize, y:0, width:Width * Scale, height:Height *
<                                 Scale)
---
>       // save the starting x value //
>       StartX = X
1385,1398c1256,1260
<       // set the board bounds //
<       BoardBounds = CGRect(x:0, y:0, width:BoardSize * Scale, height:BoardSize * Scale)
<
<       // create the gallery layer //
<       GalleryLayer = CGLayerOfSize(Width * Scale, Height:Height * Scale)
<
<       // create the diagram layer //
<       DiagramLayer = CGLayerOfSize(Width * Scale, Height:Height * Scale)
<
<       // set the diagram layer path width //
<       DiagramLayer.context?.setLineWidth(3 * Scale)
<
<       // set the diagram path line cap style to round //
<       DiagramLayer.context?.setLineCap(CGLineCap.round)
---
>       // remove the animation effects //
>       CATransaction.setDisableActions(true)
>
>       // set the animation starting point //
>       Animation.fromValue = NSValue(point:NSPoint(x:X, y:Y))
1400,1415c1262,1263
<       // create the context //
<       Context = CGContext(data:nil, width:Int(Width * Scale), height:Int(Height * Scale),
<                          bitsPerComponent:8, bytesPerRow:0, space:CGColorSpaceCreateDeviceRGB(),
<                          bitmapInfo:CGImageAlphaInfo.noneSkipFirst.rawValue)
<
<       // remove all the rectangles //
<       Rects.removeAll()
<
<       // get the number of frames that can span the width of the view //
<       FramesWide = Int(Width / FrameSize)
<
<       // update the gallery button states //
<       Board.SetGalleryButtonsToState(true)
<
<       // if is a wraparound gallery //
<       if Wraparound
---
>       // if going right //
>       if GoingRight
1417,1418c1265,1269
<          // loop over all the frames wide //
<          for Index in 0 ..< FramesWide
---
>          // decrement the x location value by one frame width //
>          X -= FrameWidth
>
>          // if went past the minimum x location value //
>          if X <= CenterX - MaxOffset
1420,1422c1271,1272
<             // append a rectangle to draw a board in //
<             Rects.append(CGRect(x:Scale * FrameSize * CGFloat(Index), y:Scale * FrameSize, width:
<                         Scale * FrameSize, height:Scale * FrameSize))
---
>             // set x to the minimum x location value //
>             X = CenterX - MaxOffset
1425,1427c1275,1276
<
<       // loop over all the frames wide //
<       for Index in 0 ..< FramesWide
---
>       // going left //
>       else
1429,1432c1278,1279
<          // append a rectangle to draw a board in //
<          Rects.append(CGRect(x:Scale * FrameSize * CGFloat(Index), y:0, width:Scale * FrameSize,
<                     height:Scale * FrameSize))
<       }
---
>          // increment the x location value by one frame width //
>          X += FrameWidth
1434,1438c1281,1282
<       // if there are no frames yet //
<       if Frames.count == 0
<       {
<          // loop over 800 frames //
<          for _ in 0 ..< 800
```

```
---
>          // if went past the maximum x location value //
>          if X >= CenterX
1440,1441c1284,1285
<             // append a frame for the given frame size //
<             Frames.append(CGLayerOfSize(BoardSize, Height:BoardSize))
---
>             // set x to the maximum x location value //
>             X = CenterX
1443,1449c1287,1293
<       }
<
<       // add a tracking rectangle for the view //
<       TrackingRectangle = addTrackingRect(bounds, owner:self, userData:nil, assumeInside:false)
<
<       // disable the cursor rectangles //
<       window!.disableCursorRects()
---
>       }
>
>       // set the animation layer position //
>       SetAnimationLayerPosition()
>
>       // add the animation to the animation layer //
>       AnimationLayer.add(Animation, forKey:"position")
2073c1917,1918
<          if NSKeyedUnarchiver.unarchiveObject(with:AppPreferences.object(forKey:Key) as! Data) != nil
---
>          if NSKeyedUnarchiver.unarchiveObject(with:AppPreferences.object(forKey:Key) as! Data) !=
>                                                  nil
2076,2077c1921,1922
<             return NSKeyedUnarchiver.unarchiveObject(with:AppPreferences.object(forKey:Key) as! Data)
<                                                         as! NSImage
---
>             return NSKeyedUnarchiver.unarchiveObject(with:AppPreferences.object(forKey:Key) as!
>                                                         Data) as! NSImage
2132,2134d1976
<    // the gallery review timer //
<    var GalleryReviewTimer:Timer = Timer()
<
2170,2172d2011
<
<    // the movie view bounds //
<    var MovieViewBounds:CGRect = CGRect(x:0, y:0, width:1, height:1)
2253c2092
<    var CGBoardSize:[CGFloat] = [180, 250, 400, 660]
---
>    var CGBoardSize:[CGFloat] = [180, 250, 660]
2256c2095
<    var CGSquareSize:[CGFloat] = [0, 0, 0, 0]
---
>    var CGSquareSize:[CGFloat] = [0, 0, 0]
2259c2098
<    var CGBoardOffset:[CGFloat] = [0, 0, 0, 0]
---
>    var CGBoardOffset:[CGFloat] = [0, 0, 0]
2404a2244,2246
>    // the number of gallery movie frames //
>    var GalleryMovieFrames:Int = 0
>
2689a2532,2534
>    // the movie size //
>    var MovieSize:Int = 1
>
3117c2962
<    // the captured piece highlight layers //
---
>    // the captured piece highlight layer //
3120c2965
<    // the threatened piece highlight layers //
---
>    // the threatened piece highlight layer //
3129c2974
<    // the last move highlight layers //
---
>    // the last move highlight layer //
3173,3175d3017
<
<    // the write gallery movie thread //
<    var WriteGalleryMovieThread:Thread = Thread()
3177c3019
<    // the name of the chess engine //
---
>    // the name of chess engine //
3353a3196,3198
>    // the gallery color //
>    var GalleryColor:NSColor = NSColor()
>
3453,3454c3298,3299
<    // the game gallery undo path button //
<    @IBOutlet var GameGalleryUndoPath:NSToolbarItem!
---
>    // the game gallery undo shape button //
>    @IBOutlet var GameGalleryUndoShape:NSToolbarItem!
3698a3544,3546
>    // the movie size slider //
>    @IBOutlet var MovieSizeSlider:NSSlider!
>
3774c3622
<    // the wraparound menu item //
---
>    // the wraparound memu item //
3821a3670,3672
>    // the text field for the movie size view //
>    @IBOutlet var MovieSizeView:NSTextField!
>
4069c3920,3921
<                           .UpdateGameClock), userInfo:nil, repeats:true), forMode:RunLoop.Mode.common)
---
>                           .UpdateGameClock), userInfo:nil, repeats:true), forMode:
>                           RunLoop.Mode.common)
4497c4349
<             return min(CGFloat(PGNMovesData.RowHeight(TableRow / 2)), CGBoardSize[3])
---
>             return min(CGFloat(PGNMovesData.RowHeight(TableRow / 2)), CGBoardSize[2])
4593,4595d4444
<          // begin a new diagram layer path //
<          DiagramLayer.context?.beginPath()
<
4851c4700,4701
<       AnimationLayer.bounds = CGRect(x:0, y:0, width:CGFloat(SquareSize), height:CGFloat(SquareSize))
---
>       AnimationLayer.bounds = CGRect(x:0, y:0, width:CGFloat(SquareSize), height:
>                                       CGFloat(SquareSize))
5401,5403c5251,5253
<       WindowMaxSize = NSMakeSize(CGFloat(660 + BoolToInt(!HideInfoPanel) * BoolToInt(SmallInfoPanel)
<                                  * 340 + BoolToInt(!HideInfoPanel) * BoolToInt(LargeInfoPanel) * 600),
<                                  660)
---
```

```
>       WindowMaxSize = NSMakeSize(CGFloat(660 + BoolToInt(!HideInfoPanel) *
>                                 BoolToInt(SmallInfoPanel) * 340 + BoolToInt(!HideInfoPanel) *
>                                 BoolToInt(LargeInfoPanel) * 600), 660)
5434c5284,5285
<          PGNMovesTableView.enclosingScrollView?.frame = NSMakeRect(692, 30, 275, 235 + HideClock * 60)
---
>          PGNMovesTableView.enclosingScrollView?.frame = NSMakeRect(692, 30, 275, 235 + HideClock *
>                                                                   60)
5481c5332
<       DatabaseExplorerWindow.setContentSize(NSMakeSize(MainWindow.frame.size.width, 30))
---
>       DatabaseExplorerWindow.setContentSize(NSMakeSize(MainWindow.frame.size.width, 25))
5492,5494c5343,5345
<       // set the gallery view frame side //
<       GameGalleryView.FrameSize = MainWindow.frame.size.width / CGFloat(Int(MainWindow.frame
<                                   .size.width / 220))
---
>       // set the gallery view frame width //
>       GameGalleryView.FrameWidth = MainWindow.frame.size.width / CGFloat(Int(MainWindow.frame
>                                    .size.width / 220))
5804c5655,5657
<             while ReadOpeningBookThread.isExecuting {}
---
>             while ReadOpeningBookThread.isExecuting
>             {
>             }
6096c5949
<          Alert.icon = NSImageFromCGLayer(3, Size:CGBoardSize[3])
---
>          Alert.icon = NSImageFromCGLayer(2, Size:CGBoardSize[2])
6344,6345c6197,6198
<       GameClock.UpdateClock(Float(ElapsedWhiteTime) / 10.0, ElapsedBlackTime:Float(ElapsedBlackTime)
<                             / 10.0)
---
>       GameClock.UpdateClock(Float(ElapsedWhiteTime) / 10.0, ElapsedBlackTime:
>                             Float(ElapsedBlackTime) / 10.0)
7030c6883,6884
<       ImageToLayer(ChessBoardImages[BoardType], Layer:CGLayerBoardType, Size:CGBoardSize[3] * Scale)
---
>       ImageToLayer(ChessBoardImages[BoardType], Layer:CGLayerBoardType, Size:CGBoardSize[2] *
>                  Scale)
7122a6977,6979
>       // set the game gallery animation rate //
>       GameGalleryView.SetAnimationRate(GalleryAnimationRate)
>
7175c7032,7033
<       BoardTintHue = PreferenceInt("BoardTintHue", Default:30, Min:0, Max:BoardTintHues.count - 1)
---
>       BoardTintHue = PreferenceInt("BoardTintHue", Default:30, Min:0, Max:BoardTintHues.count
>                                    - 1)
7184,7185c7042,7043
<       BorderTextureColor = 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)
7228a7087,7095
>       // retrieve the movie size preference //
>       MovieSize = PreferenceInt("MovieSize", Default:100, Min:50, Max:100)
>
>       // update the movie size slider //
>       MovieSizeSlider.intValue = Int32(MovieSize)
>
>       // display the movie size //
>       DisplayMovieSize()
>
7438c7305
<                                                                            "\(PreferencesSelection)")
---
>                                                                  "\(PreferencesSelection)")
8134c8001,8002
<       else if Int(Attributes.FENStartMoveNumber)! < 1 || Int(Attributes.FENStartMoveNumber)! > 1000
---
>       else if Int(Attributes.FENStartMoveNumber)! < 1 || Int(Attributes.FENStartMoveNumber)!
>                  > 1000
8259c8127,8128
<       NSBezierPath(rect:NSMakeRect(1, 1, CGFloat(SquareSize - 2), CGFloat(SquareSize - 2))).stroke()
---
>       NSBezierPath(rect:NSMakeRect(1, 1, CGFloat(SquareSize - 2), CGFloat(SquareSize - 2)))
>                   .stroke()
8265c8134,8135
<       NSBezierPath(rect:NSMakeRect(2, 2, CGFloat(SquareSize - 4), CGFloat(SquareSize - 4))).stroke()
---
>       NSBezierPath(rect:NSMakeRect(2, 2, CGFloat(SquareSize - 4), CGFloat(SquareSize - 4)))
>                   .stroke()
8272c8142
<                   CGLayerSquareBorder, Size:CGBoardSize[3] * Scale)
---
>                   CGLayerSquareBorder, Size:CGBoardSize[2] * Scale)
8274,8275c8144,8145
<       // loop over all the board size sets //
<       for Count in 0 ..< CGBoardSize.count
---
>       // loop over all three board size sets //
>       for Count in 0 ..< 3
8417c8287,8288
<       NSBezierPath(ovalIn:NSMakeRect(2, 2, CGFloat(SquareSize - 4), CGFloat(SquareSize - 4))).fill()
---
>       NSBezierPath(ovalIn:NSMakeRect(2, 2, CGFloat(SquareSize - 4), CGFloat(SquareSize - 4)))
>                   .fill()
8474c8345
<       MergeLayer(BoardBottomLayer, ToLayer:CGLayersBoard[3], Size:CGBoardSize[3] * Scale)
---
>       MergeLayer(BoardBottomLayer, ToLayer:CGLayersBoard[2], Size:CGBoardSize[2] * Scale)
8477c8348
<       MergeLayer(BoardTopLayer, ToLayer:CGLayersBoard[3], Size:CGBoardSize[3] * Scale)
---
>       MergeLayer(BoardTopLayer, ToLayer:CGLayersBoard[2], Size:CGBoardSize[2] * Scale)
8487c8358
<       UpdateBoardWithPiecesLayer(3)
---
>       UpdateBoardWithPiecesLayer(2)
8497,8500c8368
<       ViewBounds = CGRect(x:0, y:0, width:CGBoardSize[3] * Scale, height:CGBoardSize[3] * Scale)
<
<       // set the game movie view bounds //
<       MovieViewBounds = CGRect(x:0, y:0, width:CGBoardSize[2] * Scale, height:CGBoardSize[2] * Scale)
---
>       ViewBounds = CGRect(x:0, y:0, width:CGBoardSize[2] * Scale, height:CGBoardSize[2] * Scale)
8506c8374
<       CGLayerBorderOverlay = CGLayerOfSize(CGBoardSize[3] * Scale)
---
>       CGLayerBorderOverlay = CGLayerOfSize(CGBoardSize[2] * Scale)
8509c8377
<       CGLayerBoardType = CGLayerOfSize(CGBoardSize[3] * Scale)
---
>       CGLayerBoardType = CGLayerOfSize(CGBoardSize[2] * Scale)
8512c8380
<       CGLayerSquareBorder = CGLayerOfSize(CGBoardSize[3] * Scale)
---
```

```
>       CGLayerSquareBorder = CGLayerOfSize(CGBoardSize[2] * Scale)
8515c8383
<       BoardBottomLayer = CGLayerOfSize(CGBoardSize[3] * Scale)
---
>       BoardBottomLayer = CGLayerOfSize(CGBoardSize[2] * Scale)
8518c8386
<       BoardTopLayer = CGLayerOfSize(CGBoardSize[3] * Scale)
---
>       BoardTopLayer = CGLayerOfSize(CGBoardSize[2] * Scale)
8524c8392
<       DiagramLayer = CGLayerOfSize(CGBoardSize[3] * Scale)
---
>       DiagramLayer = CGLayerOfSize(CGBoardSize[2] * Scale)
8543,8544c8411,8412
<       // loop over all the board size sets //
<       for Count in 0 ..< CGBoardSize.count
---
>       // loop over all three board size sets //
>       for Count in 0 ..< 3
8547c8415
<           CGBoardOffset[Count] = 30.0 / CGBoardSize[3] * CGBoardSize[Count]
---
>           CGBoardOffset[Count] = 30.0 / CGBoardSize[2] * CGBoardSize[Count]
8586c8454
<       AppNameView.stringValue = "GreenChess 8.1.5"
---
>       AppNameView.stringValue = "GreenChess 8.1.4"
8733a8602,8604
>
>       // add the animation layer to the game gallery view //
>       GameGalleryView.layer?.addSublayer(GameGalleryView.AnimationLayer)
8765,8766c8636,8637
<       // loop over all the board size sets //
<       for Count in 0 ..< CGBoardSize.count
---
>       // loop over the three board sets //
>       for Count in 0 ..< 3
8868c8739,8740
<       NSGraphicsContext.current = NSGraphicsContext(cgContext:MovieInfoLayer.context!, flipped:false)
---
>       NSGraphicsContext.current = NSGraphicsContext(cgContext:MovieInfoLayer.context!, flipped:
>                                                     false)
8874c8746
<       NSGraphicsContext.current?.cgContext.draw(CGLayersBoardWithPieces[2], in:MovieViewBounds)
---
>       NSGraphicsContext.current?.cgContext.draw(CGLayersBoardWithPieces[2], in:ViewBounds)
8877c8749
<       MovieInfoAttributes = [NSAttributedString.Key.font:NSFont.boldSystemFont(ofSize:Scale * 11),
---
>       MovieInfoAttributes = [NSAttributedString.Key.font:NSFont.boldSystemFont(ofSize:Scale * 18),
8881c8753
<       MovieInfoTagPosition = 353
---
>       MovieInfoTagPosition = 583
8922,8924c8794,8796
<           NSBezierPath(roundedRect:(NSMakeRect(Scale * 21, Scale * (MovieInfoTagPosition - 2),
<                       AttributedMovieTag.size().width + Scale * 12, Scale * 16.5)), xRadius:Scale *
<                       8.25, yRadius:Scale * 8.25).fill()
---
>           NSBezierPath(roundedRect:(NSMakeRect(Scale * 35, Scale * (MovieInfoTagPosition - 3),
>                       AttributedMovieTag.size().width + Scale * 20, Scale * 26)), xRadius:Scale *
>                       13, yRadius:Scale * 13).fill()
8927c8799
<           AttributedMovieTag.draw(at:NSPoint(x:Scale * 27, y:Scale * MovieInfoTagPosition))
---
>           AttributedMovieTag.draw(at:NSPoint(x:Scale * 45, y:Scale * MovieInfoTagPosition))
8930c8802
<           MovieInfoTagPosition -= 45.455
---
>           MovieInfoTagPosition -= 75
9022c8894,8895
<       NSGraphicsContext.current = NSGraphicsContext(cgContext:BoardTopLayer.context!, flipped:false)
---
>       NSGraphicsContext.current = NSGraphicsContext(cgContext:BoardTopLayer.context!, flipped:
>                                                     false)
9076c8949
<       ImageToLayer(LayerImage, Layer:CGLayerBorderOverlay, Size:CGBoardSize[3] * Scale)
---
>       ImageToLayer(LayerImage, Layer:CGLayerBorderOverlay, Size:CGBoardSize[2] * Scale)
9180c9053
<       if InPositionEditor && Set == 3
---
>       if InPositionEditor && Set == 2
9194c9067,9068
<       if HighlightCapturedPieces && !ReviewingGame && !InPositionEditor && !CoreAnimating && Set != 0
---
>       if HighlightCapturedPieces && !ReviewingGame && !InPositionEditor && !CoreAnimating && Set
>           != 0
9209,9210c9083,9084
<       if HighlightThreatenedPieces && !ReviewingGame && !InPositionEditor && !CoreAnimating && Set
<           != 0
---
>       if HighlightThreatenedPieces && !ReviewingGame && !InPositionEditor && !CoreAnimating &&
>           Set != 0
9241,9242c9115,9116
<       // if move by selecting squares and have a selected square for main sized board //
<       if MoveBySelectingSquares && SelectedSquares > 0 && Set == 3
---
>       // if move by selecting squares and have a selected square //
>       if MoveBySelectingSquares && SelectedSquares > 0 && Set == 2
9275,9276c9149,9150
<       // if main sized board and not hiding captured pieces in board and not in position editor //
<       if Set == 3 && !HideCapturedPiecesInBoard && !InPositionEditor
---
>       // if set is two and not hiding captured pieces in board and not in position editor //
>       if Set == 2 && !HideCapturedPiecesInBoard && !InPositionEditor
9347c9221
<       CGContexts[3].draw(CGLayersBoard[3], in:ViewBounds)
---
>       CGContexts[2].draw(CGLayersBoard[2], in:ViewBounds)
9350c9224
<       return CGContexts[3].makeImage()!
---
>       return CGContexts[2].makeImage()!
9353c9227
<   // get a cg image from a movie info cg layer //
---
>   // get a cgimage from a movie info cg layer //
9357c9231
<       CGContexts[2].draw(MovieInfoLayer, in:MovieViewBounds)
---
>       CGContexts[2].draw(MovieInfoLayer, in:ViewBounds)
9359c9233
<       // return the cg image //
---
>       // return a cg image from the context //
9420,9421c9294,9295
<       // if the cursor location is off the board or have attached sheet //
<       if PointOutOfBounds(Cursor) || HaveAttachedSheet()
---
```

```
>        // if the mouse location is off the board or have attached sheet //
>        if Cursor.x < 30 || Cursor.x > 630 || Cursor.y < 30 || Cursor.y > 630 || HaveAttachedSheet()
9519c9393,9394
<        if PointOutOfBounds(MouseLocation)
---
>        if MouseLocation.x < 30 || MouseLocation.y < 30 || MouseLocation.x > 630 || MouseLocation.y
>           > 630
9526,9527c9401
<        if EngineThinking || CoreAnimating || (BoardGameOver(Board) && !InPositionEditor) ||
<           AnimateProgress
---
>        if EngineThinking || CoreAnimating || (BoardGameOver(Board) && !InPositionEditor)
9671,9689d9544
<     // process the diagram path //
<     func ProcessDiagramPath()
<     {
<        // keep the marker in bounds //
<        KeepMarkerInBounds()
<
<        // set the path location //
<        PathLocation = CGPoint(x:MouseLocation.x * Scale, y:MouseLocation.y * Scale)
<
<        // add a line to the diagram path location //
<        DiagramPaths[DiagramPaths.count - 1].addLine(to:PathLocation)
<
<        // add a line to the diagram layer context path //
<        DiagramLayer.context?.addLine(to:PathLocation)
<
<        // stroke the diagram layer path //
<        DiagramLayer.context?.strokePath()
<     }
<
9702,9703c9557,9570
<           // process the diagram path //
<           ProcessDiagramPath()
---
>           // keep the marker in bounds //
>           KeepMarkerInBounds()
>
>           // set the path location //
>           PathLocation = CGPoint(x:MouseLocation.x * Scale, y:MouseLocation.y * Scale)
>
>           // add a line to the diagram path location //
>           DiagramPaths[DiagramPaths.count - 1].addLine(to:PathLocation)
>
>           // add a line to the diagram layer context path //
>           DiagramLayer.context?.addLine(to:PathLocation)
>
>           // stroke the diagram layer path //
>           DiagramLayer.context?.strokePath()
9833c9700,9702
<              while ReadOpeningBookThread.isExecuting {}
---
>              while ReadOpeningBookThread.isExecuting
>              {
>              }
9918c9787,9788
<           GCEngineThread = Thread(target:self, selector:#selector(GreenChess.GCEngineMove), object:nil)
---
>           GCEngineThread = Thread(target:self, selector:#selector(GreenChess.GCEngineMove), object:
>                                   nil)
9946,9952d9815
<
<     // if the point is out of bounds //
<     func PointOutOfBounds(_ Point:NSPoint) -> Bool
<     {
<        // return if point is outside these bounds //
<        return Point.x < 30 || Point.y < 30 || Point.x > 630 || Point.y > 630
<     }
9966,9967c9829,9842
<           // process the diagram path //
<           ProcessDiagramPath()
---
>           // keep the marker in bounds //
>           KeepMarkerInBounds()
>
>           // set the path location //
>           PathLocation = CGPoint(x:MouseLocation.x * Scale, y:MouseLocation.y * Scale)
>
>           // add a line to the path location for the diagram paths //
>           DiagramPaths[DiagramPaths.count - 1].addLine(to:PathLocation)
>
>           // add a line to the path location in the diagram layer context //
>           DiagramLayer.context?.addLine(to:PathLocation)
>
>           // stroke the current path in the diagram layer context //
>           DiagramLayer.context?.strokePath()
9979,9980c9854,9856
<        // if mouse location is out of bounds and moving by selecting squares //
<        if PointOutOfBounds(MouseLocation) && MoveBySelectingSquares && SelectedSquares == 2
---
>        // if mouse location is out of range and moving by selecting squares //
>        if (MouseLocation.x < 30 || MouseLocation.y < 30 || MouseLocation.x > 630 || MouseLocation.y
>           > 630) && (MoveBySelectingSquares && SelectedSquares == 2)
10071c9947,9949
<        while GameSoundThread.isExecuting {}
---
>        while GameSoundThread.isExecuting
>        {
>        }
10141c10019,10020
<        GameSoundThread = Thread(target:self, selector:#selector(GreenChess.PlayGameSound), object:nil)
---
>        GameSoundThread = Thread(target:self, selector:#selector(GreenChess.PlayGameSound), object:
>                                 nil)
10258c10137
<        NSGraphicsContext.current?.cgContext.draw(CGLayersBoardWithPieces[3], in:bounds)
---
```

```
>       NSGraphicsContext.current?.cgContext.draw(CGLayersBoardWithPieces[2], in:bounds)
10276,10334d10154
<    // update the diagram color //
<    func UpdateDiagramColor(_ Color:String)
<    {
<       // play the game sound of a click //
<       PlayGameSoundThread("click")
<
<       // set the diagram cursor //
<       SetDiagramCursor(Color)
<
<       // if the color is cyan //
<       if Color == "cyan"
<       {
<          // set the diagram color to cyan //
<          DiagramColor = NSColor.cyan
<       }
<       // if the color is blue //
<       else if Color == "blue"
<       {
<          // set the diagram color to blue //
<          DiagramColor = NSColor.blue
<       }
<       // if the color is yellow //
<       else if Color == "yellow"
<       {
<          // set the diagram color to yellow //
<          DiagramColor = NSColor.yellow
<       }
<       // if the color is black //
<       else if Color == "black"
<       {
<          // set the diagram color to black //
<          DiagramColor = NSColor.black
<       }
<       // if the color is white //
<       else if Color == "white"
<       {
<          // set the diagram color to white //
<          DiagramColor = NSColor.white
<       }
<       // if the color is magenta //
<       else if Color == "magenta"
<       {
<          // set the diagram color to magenta //
<          DiagramColor = NSColor.magenta
<       }
<       // if the color is green //
<       else if Color == "green"
<       {
<          // set the diagram color to green //
<          DiagramColor = NSColor.green
<       }
<       // if the color to red //
<       else if Color == "red"
<       {
<          // set the diagram color to red //
<          DiagramColor = NSColor.red
<       }
<    }
<
10338,10339c10158,10165
<       // update the diagram color to blue //
<       UpdateDiagramColor("blue")
---
>       // set the diagram color to blue //
>       DiagramColor = NSColor.blue
>
>       // play the sound of a button click //
>       PlayGameSoundThread("click")
>
>       // set diagram cursor to blue highlighter //
>        // set diagram cursor to blue highlighter //
>        SetDiagramCursor("blue")
10345,10346c10171,10178
<        // update the diagram color to cyan //
<        UpdateDiagramColor("cyan")
---
>        // set the diagram color to cyan //
>        DiagramColor = NSColor.cyan
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set diagram cursor to cyan highlighter //
>        SetDiagramCursor("cyan")
10352,10353c10184,10191
<        // update the diagram color to magenta //
<        UpdateDiagramColor("magenta")
---
>        // set the diagram color to magenta //
>        DiagramColor = NSColor.magenta
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set the diagram cursor to magenta highlighter //
>        SetDiagramCursor("magenta")
10359,10360c10197,10204
<        // update the diagram color to white //
<        UpdateDiagramColor("white")
---
>        // set the diagram color to white //
>        DiagramColor = NSColor.white
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set the diagram cursor to white highlighter //
>        SetDiagramCursor("white")
10366,10367c10210,10217
<        // update the diagram color to black //
<        UpdateDiagramColor("black")
---
>        // set the diagram color to black //
>        DiagramColor = NSColor.black
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set the diagram cursor to black //
>        SetDiagramCursor("black")
10373,10374c10223,10230
<        // update the diagram color to yellow //
<        UpdateDiagramColor("yellow")
---
>        // set the diagram color to yellow //
>        DiagramColor = NSColor.yellow
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set the diagram cursor to yellow //
>        SetDiagramCursor("yellow")
10380,10381c10236,10243
<        // update the diagram color to green //
<        UpdateDiagramColor("green")
---
>        // set the diagram color to green //
>        DiagramColor = NSColor.green
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set the diagram cursor to green //
>        SetDiagramCursor("green")
10387,10388c10249,10256
<        // update the diagram color to red //
<        UpdateDiagramColor("red")
---
>        // set the diagram color to red //
>        DiagramColor = NSColor.red
>
>        // play the sound of a button click //
>        PlayGameSoundThread("click")
>
>        // set the diagram cursor to red //
>        SetDiagramCursor("red")
10391,10392c10259,10260
<    // clear the gallery diagram //
<    @IBAction func ClearGalleryDiagram(_:AnyObject)
---
>    // the erase shapes action //
>    @IBAction func EraseShapes(_:AnyObject)
10397,10398c10265,10266
<        // erase all the gallery view paths //
<        GameGalleryView.ClearDiagram()
---
>        // erase all the gallery view shapes //
>        GameGalleryView.EraseShapes()
10401,10402c10269,10270
<    // the undo gallery path action //
<    @IBAction func UndoGalleryPath(_:AnyObject)
---
>    // the undo shape action //
>    @IBAction func UndoShape(_:AnyObject)
10407,10408c10275,10276
<        // undo a path on the gallery view //
<        GameGalleryView.UndoPath()
---
>        // undo a shape on the gallery view //
>        GameGalleryView.UndoShape()
10414,10415c10282,10289
<        // update the gallery diagram color to blue //
<        GameGalleryView.UpdateGalleryColor("blue")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to blue //
>        GalleryColor = NSColor.blue
>
>        // set the gallery view diagram cursor to blue highlighter //
>        GameGalleryView.SetDiagramCursor("blue")
10421,10422c10295,10302
<        // update the gallery diagram color to cyan //
<        GameGalleryView.UpdateGalleryColor("cyan")
```

```
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to cyan //
>        GalleryColor = NSColor.cyan
>
>        // set the gallery view diagram cursor to cyan highlighter //
>        GameGalleryView.SetDiagramCursor("cyan")
10428,10429c10308,10315
<        // update the gallery diagram color to yellow //
<        GameGalleryView.UpdateGalleryColor("yellow")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to yellow //
>        GalleryColor = NSColor.yellow
>
>        // set the gallery view diagram cursor to yellow highlighter //
>        GameGalleryView.SetDiagramCursor("yellow")
10435,10436c10321,10328
<        // update the gallery diagram color to black //
<        GameGalleryView.UpdateGalleryColor("black")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to black //
>        GalleryColor = NSColor.black
>
>        // set the gallery view diagram cursor to black highlighter //
>        GameGalleryView.SetDiagramCursor("black")
10442,10443c10334,10341
<        // update the gallery diagram color to white //
<        GameGalleryView.UpdateGalleryColor("white")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to white //
>        GalleryColor = NSColor.white
>
>        // set the gallery view diagram cursor to white highlighter //
>        GameGalleryView.SetDiagramCursor("white")
10449,10450c10347,10354
<        // update the gallery diagram color to magenta //
<        GameGalleryView.UpdateGalleryColor("magenta")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to magenta //
>        GalleryColor = NSColor.magenta
>
>        // set the gallery view diagram cursor to magenta highlighter //
>        GameGalleryView.SetDiagramCursor("magenta")
10456,10457c10360,10367
<        // update the gallery diagram color to green //
<        GameGalleryView.UpdateGalleryColor("green")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to green //
>        GalleryColor = NSColor.green
>
>        // set the gallery view diagram cursor to green highlighter //
>        GameGalleryView.SetDiagramCursor("green")
10463,10464c10373,10380
<        // update the gallery diagram color to red //
<        GameGalleryView.UpdateGalleryColor("red")
---
>        // play the game sound of a click //
>        PlayGameSoundThread("click")
>
>        // set the gallery color to red //
>        GalleryColor = NSColor.red
>
>        // set the gallery view diagram cursor to red highlighter //
>        GameGalleryView.SetDiagramCursor("red")
10469a10386,10392
>        // if currently animating //
>        if GameGalleryView.Animating
>        {
>           // exit function since currently animating //
>           return
>        }
>
10498a10422,10428
>        // if currently animating //
>        if GameGalleryView.Animating
>        {
>           // exit function since currently animating //
>           return
>        }
>
10557c10487
<
---
>
10572,10576c10502,10506
<                                    UnsafeMutableRawPointer(mutating:BaseAddress!), Image.bytesPerRow,
<                                    {ReleaseContext, BaseAddress in let ContextData = Unmanaged<CFData>
<                                    .fromOpaque(ReleaseContext!); ContextData.release()},
<                                    Unmanaged<CFData>.passRetained(CGImageData!).toOpaque(), nil,
<                                    &PixelBuffer)
---
>                                    UnsafeMutableRawPointer(mutating:BaseAddress!), Image
>                                    .bytesPerRow, {ReleaseContext, BaseAddress in let ContextData =
>                                    Unmanaged<CFData>.fromOpaque(ReleaseContext!); ContextData
>                                    .release()}, Unmanaged<CFData>.passRetained(CGImageData!)
>                                    .toOpaque(), nil, &PixelBuffer)
10641,10646c10571,10572
<        // run on the main thread //
<        DispatchQueue.main.async
<        {
<           // reopen the previously attached sheet if had one //
<           self.ReopenPreviouslyAttachedSheet()
<        }
---
>        // reopen the previously attached sheet if had one //
>        ReopenPreviouslyAttachedSheet()
10652,10654d10577
<        // set the main window title to show saving movie //
<        MainWindow.title = "Saving Movie"
<
```

```
10656,10660c10579,10580
<       PrepareToWriteAMovie(GameGalleryView.Width, VidHeight:GameGalleryView.Height)
<
<       // create the write gallery movie thread //
<       WriteGalleryMovieThread = Thread(target:self, selector:#selector(GreenChess.GalleryMovieThread),
<                                        object:nil)
---
>       PrepareToWriteAMovie(GameGalleryView.Width * CGFloat(MovieSize) / 100.0, VidHeight:
>                            GameGalleryView.Height * CGFloat(MovieSize) / 100.0)
10662,10666c10582,10584
<       // set the write gallery movie thread to the highest priority //
<       WriteGalleryMovieThread.threadPriority = 1
<
<       // start the write gallery movie thread //
<       WriteGalleryMovieThread.start()
---
>       // set the number of gallery movie frames //
>       GalleryMovieFrames = Int((CGFloat(GameGalleryView.Image.width) / GameGalleryView.Scale -
>                                GameGalleryView.Width) / GameGalleryView.FrameWidth) + 1
10668,10674d10585
<       // set to animate the progress //
<       AnimateProgress = true
<    }
<
<    // the gallery movie thread //
<    @objc func GalleryMovieThread()
<    {
10676c10587
<       for FrameIndex in 0 ..< GameGalleryView.MaxPosition + 1
---
>       for FrameIndex in 0 ..< GalleryMovieFrames
10682c10593,10595
<          Adaptor!.append(PixelBufferFromCGImage(GameGalleryView.MovieFrame()),
---
>          Adaptor!.append(PixelBufferFromCGImage(GameGalleryView.Image.cropping(to:CGRect(x:
>                          GameGalleryView.FrameWidth * CGFloat(FrameIndex) * Scale, y:0, width:
>                          GameGalleryView.Width * Scale, height:GameGalleryView.Height * Scale))!),
10689,10701d10601
<
<       // run on the main thread //
<       DispatchQueue.main.async
<       {
<          // reset the main window title to gallery //
<          self.MainWindow.title = "Gallery"
<
<          // reset the gallery view //
<          self.GameGalleryView.ResetGallery()
<       }
<
<       // set animate progress to false //
<       AnimateProgress = false
10707,10711c10607,10613
<       // write the gallery image to a jpeg file //
<       try? GameGalleryView.GallerySnapshot().representation(using:NSBitmapImageRep.FileType.jpeg,
<                                                             properties:[:])!.write(to:
<                                                             URL(fileURLWithPath:Filename), options
<                                                             :[.atomic])
---
>       // take a snapshot of the gallery view //
>       GameGalleryView.TakeSnapshot()
>
>       // write the image to a jpeg file //
>       try? NSBitmapImageRep(cgImage:GameGalleryView.SnapshotBitmap.cgImage!).representation(using:
>                             NSBitmapImageRep.FileType.jpeg, properties:[:])!.write(to:
>                             URL(fileURLWithPath:Filename), options:[.atomic])
10743c10645,10646
<       PrepareToWriteAMovie(CGBoardSize[2], VidHeight:CGBoardSize[2])
---
>       PrepareToWriteAMovie(frame.size.width * CGFloat(MovieSize) / 100.0, VidHeight:
>                            frame.size.height * CGFloat(MovieSize) / 100.0)
10794c10697
<       CGContexts[3].draw(CGLayersBoardWithPieces[3], in:ViewBounds)
---
>       CGContexts[2].draw(CGLayersBoardWithPieces[2], in:ViewBounds)
10800c10703
<          CGContexts[3].draw(DiagramLayer, in:ViewBounds)
---
>          CGContexts[2].draw(DiagramLayer, in:ViewBounds)
10804c10707
<       SnapshotImageRep = NSBitmapImageRep(cgImage:CGContexts[3].makeImage()!)
---
>       SnapshotImageRep = NSBitmapImageRep(cgImage:CGContexts[2].makeImage()!)
10811c10714,10715
<                                               .write(to:URL(fileURLWithPath:Filename), options:[.atomic])
---
>                                               .write(to:URL(fileURLWithPath:Filename), options:
>                                               [.atomic])
12178,12183c12082,12087
<
<       // stop the gallery review timer //
<       GalleryReviewTimer.invalidate()
<
<       // erase the game gallery view //
<       GameGalleryView.EraseGallery()
---
>
>       // remove the tracking rectangle from the gallery view //
>       GameGalleryView.removeTrackingRect(GameGalleryView.TrackingRectangle)
>
>       // stop the gallery view animation //
>       GameGalleryView.StopAnimation()
12192a12097,12105
>
>       // erase all the gallery view shapes //
>       GameGalleryView.EraseShapes()
>
>       // reset the gallery view image to empty //
>       GameGalleryView.Image = GameGalleryView.Context.makeImage()!
>
>       // set the gallery view animation layer contents to empty //
>       GameGalleryView.AnimationLayer.contents = GameGalleryView.Image
12361a12275,12277
>
>       // update the user preference for the movie size //
>       AppPreferences.set(MovieSize, forKey:"MovieSize")
12376,12382c12292,12298
<       // loop over the first three board sizes //
<       for Set in 0 ... 2
<       {
<          // draw the board image into the given context //
<          CGLayersBoard[Set].context?.draw(CGImageFromBoardLayer(), in:CGRect(x:0, y:0, width:
<                                           CGBoardSize[Set] * Scale, height:CGBoardSize[Set] * Scale))
<       }
```

```
---
>       // draw the board image into the small context //
>       CGLayersBoard[0].context?.draw(CGImageFromBoardLayer(), in:CGRect(x:0, y:0, width:
>                                      CGBoardSize[0] * Scale, height:CGBoardSize[0] * Scale))
>
>       // draw the board image into the medium context //
>       CGLayersBoard[1].context?.draw(CGImageFromBoardLayer(), in:CGRect(x:0, y:0, width:
>                                      CGBoardSize[1] * Scale, height:CGBoardSize[1] * Scale))
12389,12390c12305,12306
<       return TextField.replacingOccurrences(of:"\"", with:"").replacingOccurrences(of:"\n",with:" ")
<                         .trimmingCharacters(in:CharacterSet.whitespacesAndNewlines)
---
>       return TextField.replacingOccurrences(of:"\"", with:"").replacingOccurrences(of:"\n",
>                         with:" ").trimmingCharacters(in:CharacterSet.whitespacesAndNewlines)
12513,12521c12429,12430
<       // if have selected a new position in the game gallery //
<       if Int(GameGallerySlider.intValue) != GameGalleryView.CurrentPosition
<       {
<          // set the current position to the selected one from the game gallery slider //
<          GameGalleryView.CurrentPosition = Int(GameGallerySlider.intValue)
<
<          // update the game gallery view //
<          GameGalleryView.UpdateGallery()
<       }
---
>       // update the game gallery view //
>       GameGalleryView.UpdateGalleryView(CGFloat(GameGallerySlider.doubleValue))
12532,12533c12441,12442
<                                        .FrameSize * CGFloat(BoolToInt(GameGalleryView.Wraparound) + 1)
<                                        + 25))
---
>                                        .FrameWidth * CGFloat(BoolToInt(GameGalleryView.Wraparound)
>                                        + 1) + 25))
12537,12538c12446,12447
<                                          .FrameSize * CGFloat(BoolToInt(GameGalleryView.Wraparound) +
<                                          1))
---
>                                          .FrameWidth * CGFloat(BoolToInt(GameGalleryView.Wraparound)
>                                          + 1))
12548a12458,12460
>       // set the game gallery slider maximum value //
>       GameGallerySlider.maxValue = 1
>
12550c12462
<       GameGallerySlider.intValue = 0
---
>       GameGallerySlider.doubleValue = 0
12557,12559d12468
<
<       // initialize the game gallery view //
<       GameGalleryView.Initialize(CGBoardSize[1])
12560a12470,12472
>       // create the game gallery image //
>       GameGalleryView.CreateGalleryImage(FrameCount:MoveList.count + 1)
>
12564,12568c12476,12480
<       // update the game gallery slider maximum value //
<       GameGallerySlider.maxValue = Double(GameGalleryView.MaxPosition)
<
<       // update the gallery //
<       GameGalleryView.UpdateGallery()
---
>       // finish up the gallery image //
>       GameGalleryView.FinishUpGalleryImage()
>
>       // update the game gallery view //
>       GameGalleryView.UpdateGalleryView(CGFloat(GameGallerySlider.doubleValue))
12667c12579
<          ImageToLayer(ChessBoardImages[BoardType], Layer:CGLayerBoardType, Size:CGBoardSize[3] *
---
>          ImageToLayer(ChessBoardImages[BoardType], Layer:CGLayerBoardType, Size:CGBoardSize[2] *
13211a13124,13129
>       // update the movie size slider frame //
>       MovieSizeSlider.frame = NSMakeRect(30, 585, 600, 13)
>
>       // update the movie size text view frame //
>       MovieSizeView.frame = NSMakeRect(30, 567, 600, 20)
>
13213c13131
<       MovieFPSSlider.frame = NSMakeRect(30, 585, 600, 13)
---
>       MovieFPSSlider.frame = NSMakeRect(30, 535, 600, 13)
13216c13134
<       MovieFPSView.frame = NSMakeRect(30, 567, 600, 20)
---
>       MovieFPSView.frame = NSMakeRect(30, 517, 600, 20)
13268a13187,13192
>       // update the movie size slider state //
>       MovieSizeSlider.isHidden = State
>
>       // update the movie size view state //
>       MovieSizeView.isHidden = State
>
13368c13292,13293
<                               FontState:true, BackgroundState:true, BoardState:true, MovieState:true)
---
>                               FontState:true, BackgroundState:true, BoardState:true, MovieState:
>                               true)
13376c13301,13302
<                               FontState:true, BackgroundState:true, BoardState:true, MovieState:true)
---
>                               FontState:true, BackgroundState:true, BoardState:true, MovieState:
>                               true)
13392c13318,13319
<                               FontState:true, BackgroundState:true, BoardState:true, MovieState:false)
---
>                               FontState:true, BackgroundState:true, BoardState:true, MovieState:
>                               false)
13400c13327,13328
<                               FontState:true, BackgroundState:true, BoardState:false, MovieState:true)
---
>                               FontState:true, BackgroundState:true, BoardState:false, MovieState:
>                               true)
13408c13336,13337
<                               FontState:true, BackgroundState:false, BoardState:true, MovieState:true)
---
>                               FontState:true, BackgroundState:false, BoardState:true, MovieState:
>                               true)
13416c13345,13346
<                               FontState:false, BackgroundState:true, BoardState:true, MovieState:true)
---
>                               FontState:false, BackgroundState:true, BoardState:true, MovieState:
>                               true)
13424,13425c13354,13355
<                            #selector(GreenChess.ThePreferencesMenuItemAction), userInfo:nil, repeats:
<                            false), forMode:RunLoop.Mode.common)
---
>                            #selector(GreenChess.ThePreferencesMenuItemAction), userInfo:nil,
>                            repeats:false), forMode:RunLoop.Mode.common)
13824c13754
<       Attributes.PositionImage = NSImageFromCGLayer(3, Size:CGBoardSize[3])
```

```
---
>       Attributes.PositionImage = NSImageFromCGLayer(2, Size:CGBoardSize[2])
13836c13766
<       SavedPositionImages.append(NSImageFromCGLayer(3, Size:CGBoardSize[3]))
---
>       SavedPositionImages.append(NSImageFromCGLayer(2, Size:CGBoardSize[2]))
14923a14854,14856
>
>          // set the game gallery view animation rate //
>          GameGalleryView.SetAnimationRate(GalleryAnimationRate)
14996a14930,14943
>    // update the movie size //
>    @IBAction func UpdateMovieSize(_:AnyObject)
>    {
>       // if the movie size is different //
>       if MovieSizeSlider.intValue != MovieSize
>       {
>          // get the movie size from the movie size slider //
>          MovieSize = Int(MovieSizeSlider.intValue)
>
>          // display the movie size //
>          DisplayMovieSize()
>       }
>    }
>
15005a14953,14959
>
>    // display the movie size //
>    func DisplayMovieSize()
>    {
>       // set the movie size view to show the movie size //
>       MovieSizeView.stringValue = "Movie size is \(MovieSize)%"
>    }
15834c15788,15789
<          PGNMovesTableView.scrollRowToVisible(min(PGNMovesList.count * 2, PGNMovesData.MaxRows) - 1)
---
>          PGNMovesTableView.scrollRowToVisible(min(PGNMovesList.count * 2, PGNMovesData.MaxRows) -
>                                               1)
16441,16442c16396,16407
<       // go left one frame in the game gallery //
<       GameGalleryView.GoLeft()
---
>       // if currently animating //
>       if GameGalleryView.Animating
>       {
>          // stop the animation //
>          GameGalleryView.StopAnimation()
>       }
>       // not currently animating //
>       else
>       {
>          // go left one frame in the game gallery //
>          GameGalleryView.Animate(false)
>       }
16448,16449c16413,16424
<       // go right one frame in the game gallery //
<       GameGalleryView.GoRight()
---
>       // if currently animating //
>       if GameGalleryView.Animating
>       {
>          // stop the animation //
>          GameGalleryView.StopAnimation()
>       }
>       // not currently animating //
>       else
>       {
>          // go right one frame in the game gallery //
>          GameGalleryView.Animate(true)
>       }
16537,16538c16512,16513
<       // update the game gallery undo path button state //
<       GameGalleryUndoPath.isEnabled = State
---
>       // update the game gallery undo shape button state //
>       GameGalleryUndoShape.isEnabled = State
16557,16563d16531
<    // go right in the game gallery for the game review //
<    @objc func GoRight()
<    {
<       // go right in the game gallery for the game review //
<       GameGalleryView.GoRight()
<    }
<
16567,16571c16535,16536
<       // invalidate the gallery diagram path //
<       GameGalleryView.ValidPath = false
<
<       // if the game gallery review maximum position is zero //
<       if GameGalleryView.MaxPosition == 0
---
>       // if animating and not reviewing //
>       if !GameGalleryView.Reviewing && GameGalleryView.Animating
16573c16538,16541
<          // return since there is nothing to do a gallery review on //
---
>          // stop the gallery view animation //
>          GameGalleryView.StopAnimation()
>
>          // exit function //
16576,16577c16544,16551
<
<       // if currently reviewing the game gallery //
---
>
>       // switch the gallery view reviewing state //
>       GameGalleryView.Reviewing = !GameGalleryView.Reviewing
>
>       // stop the gallery animation //
>       GameGalleryView.StopAnimation()
>
>       // if currently reviewing the gallery //
16580,16599c16554,16555
<          // stop the game gallery review timer //
<          GalleryReviewTimer.invalidate()
<
<          // set that no longer reviewing the game gallery //
<          GameGalleryView.Reviewing = false
<       }
<       // not currently reviewing the game gallery //
<       else
<       {
<          // create the gallery review timer //
<          GalleryReviewTimer = Timer.scheduledTimer(timeInterval:0.5 + Double(100 -
<                                                    GalleryAnimationRate) / 35.0, target:self,
<                                                    selector:#selector(GreenChess.GoRight),
<                                                    userInfo:nil, repeats:true)
<
<          // add the gallery review timer to the main run loop //
<          RunLoop.current.add(GalleryReviewTimer, forMode:RunLoop.Mode.common)
<
```

```
<              // set that currently reviewing the game gallery //
<              GameGalleryView.Reviewing = true
---
>              // go right in the game gallery view //
>              GameGalleryRightAction(self)
16601,16606d16556
<
<           // update the game gallery view cursor //
<           GameGalleryView.UpdateCursor()
<
<           // update the gallery buttons states //
<           SetGalleryButtonsToState(!GameGalleryView.Reviewing)
16762c16712,16713
<           UpdateCastleItemLabel(WhiteKingsideCastleToolbarButton, On:WhiteKingsideCastle, Kingside:true)
---
>           UpdateCastleItemLabel(WhiteKingsideCastleToolbarButton, On:WhiteKingsideCastle, Kingside:
>                                   true)
16769c16720,16721
<           UpdateCastleItemLabel(BlackKingsideCastleToolbarButton, On:BlackKingsideCastle, Kingside:true)
---
>           UpdateCastleItemLabel(BlackKingsideCastleToolbarButton, On:BlackKingsideCastle, Kingside:
>                                   true)
17189c17141
<              GameGalleryView.AddFrame(FrameNumber, Layer:CGLayersBoardWithPieces[1])
---
>              GameGalleryView.AddFrame(FrameNumber)
17391c17343
<              // update the pieces layer for the annotation/gallery size //
---
>              // update the pieces layer for the annotation size //
17394,17400c17346
<           // if recording a movie //
<           else if RecordingMovie
<           {
<              // update the board with pieces layer for the game movie board size //
<              UpdateBoardWithPiecesLayer(2)
<           }
<           // not in the game gallery, recording game movie, or annotating //
---
>           // not in the game gallery or annotating //
17404c17350
<              UpdateBoardWithPiecesLayer(3)
---
>              UpdateBoardWithPiecesLayer(2)
17406,17407c17352,17357
<              // set that need to update the display //
<              displayIgnoringOpacity(bounds)
---
>              // if not recording a movie //
>              if !RecordingMovie
>              {
>                 // set that need to update the display //
>                 displayIgnoringOpacity(bounds)
>              }
```