```
[0]
[0]
[0]
```

```
>
>            // add the pgn label to the pgn move //
>            PGNMove += " \(PGNLabel)"
>
>            // if not in the stockfish analysis move list //
>            if MoveIndex < MoveList.count
6280,6283c6566,6573
<               // replace the code occurrences with the corresponding symbols //
<               AnnotationString = AnnotationString.replacingOccurrences(of:GlyphCode[GlyphIndex],
<                                                                        with:GlyphSymbol[GlyphIndex])
<            }
---
>               // get the current move annotation string //
>               AnnotationString = AnnotationList[MoveIndex]
>
>               // convert all the numeric annotation glyphs //
>               ConvertNumericAnnotationGlyphs()
>
>               // append the pgn move to the pgn moves list for the annotator //
>               ShortPGNList.append(PGNMove)
6285,6286c6575,6576
<            // add the corresponding annotation to the pgn move string //
<            PGNMove += " " + AnnotationString
---
>               // add the corresponding annotation to the pgn move string //
>               PGNMove += " " + AnnotationString
6288,6289c6578,6586
<            // add the pgn move string to the pgn move list array //
<            PGNMovesList.append(PGNMove)
---
>               // add the pgn move string to the pgn move list array //
>               PGNMovesList.append(PGNMove)
>            }
>            // if not at the beginning of a new white/black move sequence or at end //
>            else
>            {
>               // add the pgn move to the stockfish principal variation string //
>               StockfishPV += "(" + PGNMove + ") "
>            }
6291c6588
<         // if not at the beginning of a new white/black move sequence or at end //
---
>         // the move does not exist //
6294,6295c6591,6598
<            // add the pgn move to the stockfish principal variation string //
<            StockfishPV += "(" + PGNMove + ") "
---
>            // if in the stockfish analysis move list portion, break //
>            if MoveIndex >= MoveList.count {break}
>
>            // set the pgn move string to invalid move //
>            PGNMove = "Invalid Move"
>
>            // add the pgn move string to the pgn move list array //
>            PGNMovesList.append(PGNMove)
6306,6308c6609,6611
<       // resent the stockish analysis string to empty //
<       StockfishPV = ""
<
---
>       // clear the stockfish analysis move list //
>       StockfishMoveList.removeAll()
>
6550,6551c6853,6854
<       // if the annotator slider value greater than zero //
<       if GameAnnotatorSlider.intValue > 0
---
>       // if the annotator slider value greater than one //
>       if GameAnnotatorSlider.intValue > 1
6998c7301,7311
<       Attributes.UndoToPreviousMove()
---
>       // if there are no moves in the move list to undo, return //
>       if MoveList.isEmpty {return}
>
>       // set the previous move list to the current move list //
>       PreviousMoveList = MoveList
>
>       // set the previous annotation list to the current annotation list //
>       PreviousAnnotationList = AnnotationList
>
>       // remove the last move from the move list //
>       _ = MoveList.popLast()
6999a7313,7315
>       // remove the last annotation from the move annotation list //
>       _ = AnnotationList.popLast()
>
7026c7342
<       for _ in 0...Attributes.GetCount()
---
>       for _ in 0...MoveList.count
7055,7090d7370
<    // remove the invalid moves and update the pgn notation //
<    func RemoveInvalidMovesAndUpdateNotation(_ Type:String)
<    {
<       // initialize the board //
<       InitializeBoardAttributes(Board)
<
<       // loop over all the moves in the move list including analysis moves //
<       for Index in 0 ..< Attributes.AnalysisCount()
<       {
<          // make the move on the board //
<          MakeMoveOnBoard(Attributes.GetMove(Index))
<
<          // if the move exists //
<          if MoveExists
<          {
<             // save the pgn notation //
<             Attributes.SaveNotation(Index, Notation:String(validatingUTF8:BoardPGNLabel(Board))!)
<          }
<          // the move doesn't exist //
<          else
<          {
<             // mark the move an invalid //
<             Attributes.MarkMoveInvalid(Index)
<          }
<       }
<
<       // compress the moves based on the type //
<       Attributes.Compress(Type)
<
<       // initialize the board //
<       InitializeBoardAttributes(Board)
<
<       // loop over all the moves and make them on the board //
<       for Index in 0 ..< Attributes.GetCount() {MakeMoveOnBoard(Attributes.GetMove(Index))}
<    }
<
7093a7374,7376
>       // set that initially haven't found an invalid move //
>       FoundInvalidMove = false
>
7098c7381
<       for Index in 0 ..< Attributes.GetCount()
---
>       for Index in 0 ..< MoveList.count
7101c7384
<          MovingPiece.SaveUCIMove(Attributes.GetMove(Index))
---
>          MovingPiece.SaveUCIMove(MoveList[Index])
7106,7107c7389,7393
<          // process the current uci move //
<          ProcessUCIMove()
---
>          // if the move looking for does exist //
>          if MoveExists
>          {
>             // process the current uci move //
>             ProcessUCIMove()
7109,7110c7395,7396
<          // update pawn to the specified piece if needed //
<          MovingPiece.UpdateToPromotionType()
---
>             // update pawn to the specified piece if needed //
>             MovingPiece.UpdateToPromotionType()
7112,7113c7398,7399
<          // put the piece that moved onto the destination square //
<          UpdateDestinationSquare()
---
>             // put the piece that moved onto the destination square //
>             UpdateDestinationSquare()
7115,7116c7401,7402
<          // perform castling and enpassant move if had one //
<          PerformCastleEnpassantIfNeeded()
---
>             // perform castling and enpassant move if had one //
>             PerformCastleEnpassantIfNeeded()
7118,7119c7404,7415
<          // update the illustration data //
<          UpdateIllustrationData(Index + 1)
---
>             // update the illustration data //
>             UpdateIllustrationData(Index + 1)
>          }
>          // the move doesn't exist //
>          else
>          {
>             // set the move to be an invalid move since it doesn't exist //
>             MoveList[Index] = "Invalid Move"
>
>             // set that found an invalid move //
>             FoundInvalidMove = true
>          }
7135c7431
<                         UpdateBoard:false)
---
>                         UpdateBoard:false, EnteringPreferences:EnteringPreferences)
7159,7160c7455,7457
<                         Attributes.GetV("FENEnpassantSquare"), EnpassantSquare:EnpassantSquare,
<                         RankAndFile:MovingPiece.Location, UpdateBoard:UpdateBoard)
---
>                         Attributes.FENEnpassantSquare, EnpassantSquare:EnpassantSquare, RankAndFile:
>                         MovingPiece.Location, UpdateBoard:UpdateBoard, EnteringPreferences:
>                         EnteringPreferences)
```