```
////////////////////////////////////////////////////////////////////////////////
// Swift source code of UserInterface.swift Version 9.0.6 at greenchess.com //
////////////////////////////////////////////////////////////////////////////////

// import needed system files //
import SwiftUI
import AppKit
import AVFoundation

// the observable object values //
class Values:ObservableObject
{
   // the published value //
   @Published var Val:Double = 0

   // the published minimum value //
   @Published var MinV:Double = 0

   // the published maximum value //
   @Published var MaxV:Double = 1

   // the identifier for the control //
   var Identifier:String = ""

   // the x position of the control view //
   var XP:Double = 0

   // the y position of the control view //
   var YP:Double = 0

   // the width of the control view //
   var Width:Double = 0

   // the height of the control view //
   var Height:Double = 0

   // the action for the control to perform //
   var ActionFunc:((String, Double) -> ())!

   // the text for the control to display //
   var TextFunc:((String, Double) -> (String))!

   // the values initializer //
   init(_ Val:Int, MinV:Double, MaxV:Double, XP:Double, YP:Double, Width:Double, Height:Double)
   {
      // initialize the value, min, max, x position, y position, width, and height of control //
      (self.Val, self.MinV, self.MaxV, self.XP, self.YP, self.Width, self.Height) = (Double(Val),
      MinV, MaxV, XP, YP, Width, Height)
   }

   // add the identifier and functions to the control //
   func AddFunctions(_ Identifier:String, ActionFunc:@escaping(String, Double) -> (), TextFunc:
                     @escaping(String, Double) -> String)
   {
      // set the identifier, the action function, and the text function //
      (self.Identifier, self.ActionFunc, self.TextFunc) = (Identifier, ActionFunc, TextFunc)
   }

   // the default initializer //
   init() {}
}

// a slider view structure //
struct SliderStruct:View
{
   // the observed object values //
   @ObservedObject var Vals:Values = Values()

   // the slider view initializer //
   init(_ Vals:Values) {self.Vals = Vals}

   // the swiftui view //
   var body:some View {VStack(alignment:.leading, spacing:0)
   {
      // the slider for the view and the associated function called when the value changes //
      Slider(value:Binding(get:{Vals.Val}, set:{Value in Vals.Val = Value; Vals.ActionFunc(Vals
            .Identifier, Vals.Val)}), in:Vals.MinV...Vals.MaxV).shadow(color:.gray, radius:0, x:0,
            y:0).controlSize(ControlSize.small)

      // the text to display to the view for when the slider value changes //
      Text(Vals.TextFunc(Vals.Identifier, Vals.Val)).font(.system(size:15))
   }.frame(width:Vals.Width, height:Vals.Height).position(x:Vals.XP, y:Vals.YP)}
}

// the class for a user interface object //
class UIObject
{
   // the user interface object values //
   var Vals:Values!

   // the display view //
   var TheView:NSView!

   // the window to display the view on //
   var Window:NSWindow!

   // the sub window identifier //
   var SubWindow:String = ""

   // the control type //
   var ControlType:String = ""

   // add the identifier and functions to the view structure //
   func AddFunctions(_ Identifier:String, Window:NSWindow, ActionFunc:@escaping(String, Double) ->
                     (), TextFunc:@escaping(String, Double) -> String)
   {
      // set the window to display the view on //
      self.Window = Window

      // add the identifier and functions to the values //
      Vals.AddFunctions(Identifier, ActionFunc:ActionFunc, TextFunc:TextFunc)

      // switch over the control type //
      switch ControlType
      {
         // create the view to display for a slider //
         case "Slider" : TheView = NSHostingView(rootView:SliderStruct(Vals))

         // the default case //
         default : break
      }
   }

   // the user interface object initializer //
   init(_ ControlType:String, SubWindow:String, Val:Int, MinV:Double, MaxV:Double, XP:Double, YP:
      Double, Width:Double, Height:Double)
   {
      // initialize the subwindow, control type, view value, view limits, and view bounds //
      (self.SubWindow, self.ControlType, Vals) = (SubWindow, ControlType, Values(Val, MinV:MinV,
      MaxV:MaxV, XP:XP, YP:YP, Width:Width, Height:Height))
   }
}

// the user interface class //
class UserInterface
{
   // the main view obejct //
   var GC:GreenChess!

   // a dictionary of user interface objects //
   var UIObjects:[String:UIObject] = [:]

   // the action function for the view control //
   func ActionFunc(_ Identifier:String, Value:Double)
   {
      // switch over the identifier //
      switch Identifier
      {
         // the movie frames per second from slider function to perform //
         case "MovieFPSFromSlider" : GC.FramesMovie.UpdateMovieFPS(Value)

         // the default case //
         default : break
      }
   }

   // the text display function for the view control //
   func TextFunc(_ Identifier:String, Value:Double) -> String
   {
      // switch over the identifier //
      switch Identifier
      {
         // the movie frames per second from slider text display function to perform //
         case "MovieFPSFromSlider" : return GC.FramesMovie.DisplayMovieFPS(Value)

         // the default case //
         default : return ""
      }
   }

   // get the window to display the view on //
   func GetWindow(_ Identifier:String) -> NSWindow
   {
      // switch over the identifier //
      switch Identifier
      {
         // the movie frames per second from slider window to display on //
         case "MovieFPSFromSlider" : return GC.PreferencesWindow

         // return the main window as the default //
         default : return GC.MainWindow
      }
   }

   // hide or unhide the user interface objects //
   func ControlsVisible(_ Visible:Bool, SubWindow:String)
   {
      // loop over all the values in the user interface objects dictionary //
      for (_, Value) in UIObjects {if Value.SubWindow == SubWindow
      {
         // add the view to the window if is to be visible //
         if Visible {Value.Window.contentView!.addSubview(Value.TheView)}

         // remove the view from the super view if is not to be visible //
         if !Visible {Value.TheView.removeFromSuperview()}
      }}
   }

   // initialize the user interface class //
   func Initialize(_ GC:GreenChess)
   {
      // set the main view object //
      self.GC = GC

      // add the movie frames from slider to the user interface dictionary //
      UIObjects.updateValue(UIObject("Slider", SubWindow:"PrefsMovie", Val:GC.FramesMovie.Inf
                .Vals("MovieFPSFromSlider").VI, MinV:0, MaxV:100, XP:330, YP:-600, Width:600,
                Height:30), forKey:"MovieFPSFromSlider")

      // loop over all the keys and values in the user interface objects dictionary //
      for (Key, Value) in UIObjects
      {
         // add the identifier, window, action, and text functions to the user interface object //
         Value.AddFunctions(Key, Window:GetWindow(Key), ActionFunc:ActionFunc, TextFunc:TextFunc)
      }
   }
}
```