Monkeylogic eye calibration

* + From the timing script we can access the calibration data from: **eye\_.CalFun.tform**, which is a (1,3) cell array of structures containing the useful fields:
    - **{1}:** offset
      * [x,y] Offset from the center of the screen
    - **{2}:** rotation\_t
      * [2x2] rotation matrix, usually = eye(2)
    - **{3}:** tdata.T
      * [3x3] forward projective matrix to convert eyelink raw data to degrees
  + To convert from eyelink raw data to pixel values we need these formulas:

[X, Y, adjust] = [RawX, RawY, 1] \* eye\_CalFun.tform{3}.tdata.T

EyeDegX = X / adjust

EyeDegY = Y / adjust

([EyeDegX, EyeDegY] – offset) \* rotation\_t

ScreenWidth = eye\_.Screen.Xsize

ScreenHeight = eye\_.Screen.YSize

PixPerDeg = eye\_.Screen.PixelsPerDegree

* + Assuming that the top-left is (0,0) in pixel values:
    - EyePixX = (EyeDegX \* PixPerDeg) + 0.5 \* ScreenWidth
    - EyePixY = - (EyeDegY \* PixPerDeg) + 0.5 \* ScreenHeight
  + These values are also accessible via:
    - MLConfig.EyeTransform{1,1} : offset
    - MLConfig.EyeTransform{1,2} : rotation
    - MLConfig.EyeTransform{1,3} : tform
    - MLConfig.PixelsPerDegree(1)
    - MLConfig.Screen.Xsize/Ysize

State system notes:

* The order of execution is Update -> State Machine Update -> Late Update so we send the Player/Eye data on Update to the experiment controller, then the state system checks the values on state machine update and triggers the state change accordingly.
  + On the same frame that the condition is met in Update, both the on state exit and on state enter of the next state are executed. The next frame will execute the Update of the second state.
* IMPORTANT:
  + You need to uncheck the “Has exit time” property in the animator states, if not it will add a delay between script execution and state changes. Without it it’s 1 frame, can be >10 if checked.