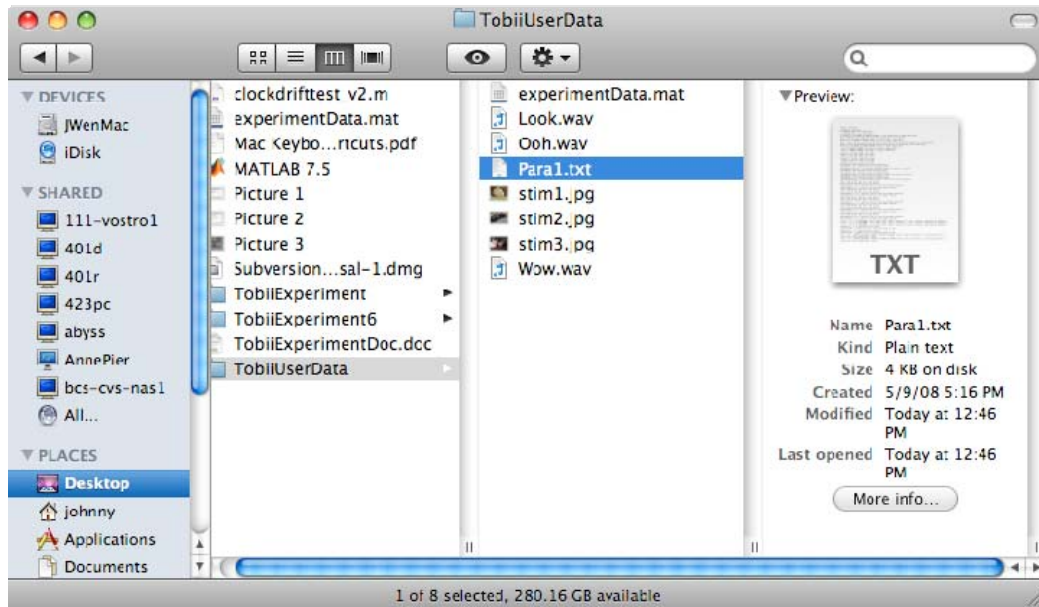


Run Experiment Procedures

Step 1) Create a folder



You can create a folder for special experiment that includes all the resource such as pictures, sound files and parameter file. After experiment, all data will also save to this folder by default. In this example, I created a folder “TobiiUserData” on Desktop.

Step 2) Open Command Window

We only can use command window to run the experiment, because we need use **talk2tobii** mex function that doesn't support MATLAB GUI.

You can use either **Terminal** or **xterm** command window.

Step 3) Run experiment

- i) start MATLAB without java GUI support.
`>> /Applications/MATLAB_R2007b/bin/matlab -nojvm`
- ii) change current directory to your created folder
`>> cd /Users/aslinlab/Desktop/TobiiUserData`
- iii) run your experiment by parameter file that was setup by TobiiGUI
`>> TobiiCommand('Para1.txt')` **Or**
`>> TobiiCommand('/Users/aslinlab/Desktop/TobiiUserData/Para1.txt')`

Notice: Do remember to set **Debug Mode Off** and **ConnTobii On** in parameter file.

E.g Para1.txt

Debug 0 '1/0=On/Off'

ConnTobii 1 '1/0=On/Off'

```
Terminal — MATLAB — 80x24
Last login: Mon May 12 15:40:52 on ttyp4
Welcome to Darwin!
Munchkin:~ aslinlab$ pwd
/Users/aslinlab
Munchkin:~ aslinlab$ /Applications/MATLAB_R2007b/bin/matlab -nojvm
Warning: No display specified. You will not be able to display graphics on the
screen.

      < M A T L A B >
    Copyright 1984-2007 The MathWorks, Inc.
      Version 7.5.0.338 (R2007b)
      August 9, 2007

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> cd /Users/aslinlab/Desktop/TobiiUserData/
>> ls
Look.wav          Wow.wav          stim2.jpg
Ooh.wav          experimentData.mat  stim3.jpg
Para1.txt        stim1.jpg

>> TobiiCommand('Para1.txt') █
```

Useful command:

- 1) pwd – show current directory
- 2) ls – show current folder contents
- 3) cd – change directory
- 4) quit – quit MATLAB program

Step 4) Control the experiment

- 1) key 's' – star and stop the image moving
- 2) key 'esc' – quit the experiment (you may need hit the 'esc' key few times)

Step 5) Save experiment data

- 1) After experiment, you can find data file in the current directory in the format of data(year/month/day/hour/minute/second).mat, e.g. **data090219134711.mat**
- 2) If the experiment crash in the middle, we still save all the trails before crash. However, the temporary data file name will be **experimentData.mat** file in the current directory and all others trial data files inside **TempTrialData** dir. After you restart your computer, you need go to same dir before crash and run command **SaveAllData**. You will get the file dataYYMMDDHHMMSS.mat back.
- 3) After you finish your experiment, you can rename the data file. Next experiment will overwrite the temporary file **experimentData.mat**.

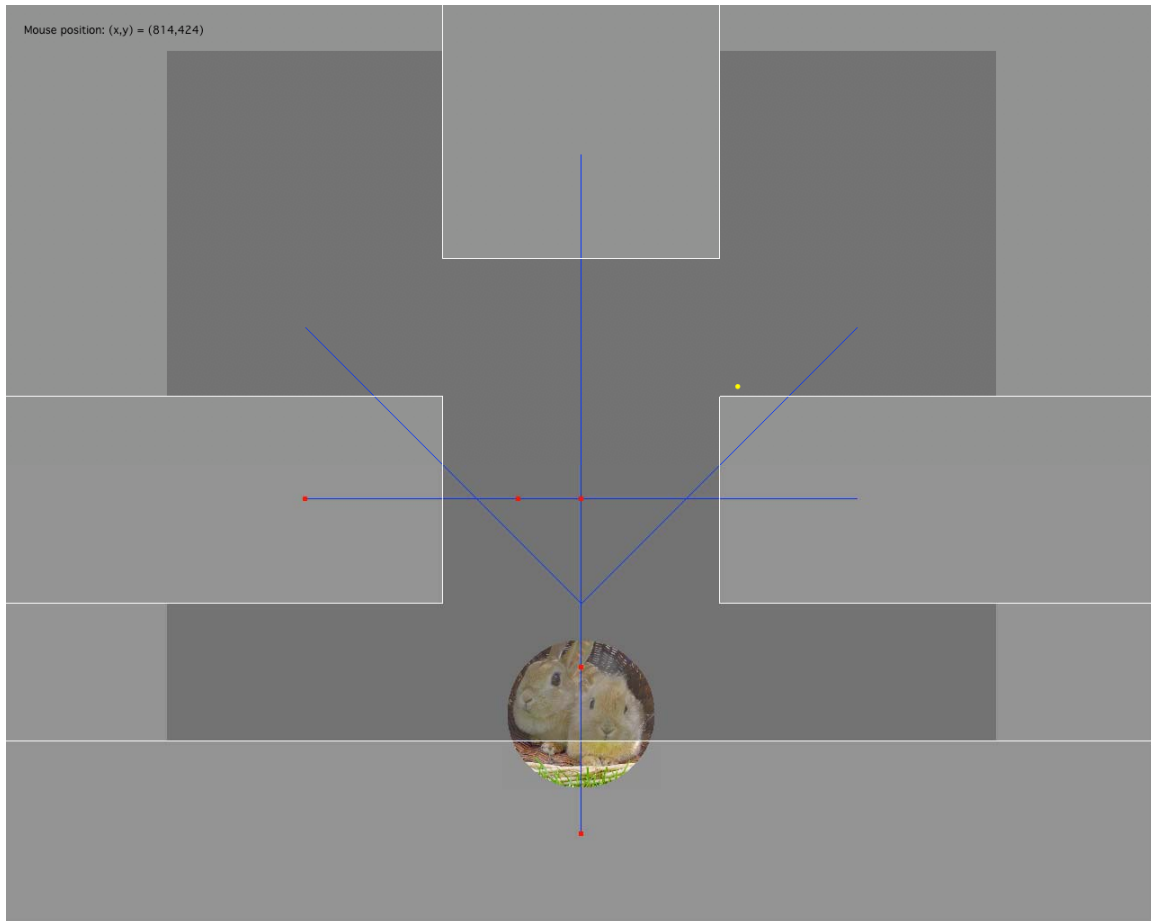
Notice: If the connection between Mac and Tobii is failure, then the experiment will not start and you will see error message in command window.

??? Error using ==> InitTobii at 29
Initialize Tobii fail!

Eye Calibration and Debug Mode

We still have problem to use **talk2tobii** mex function to do the eye calibration. Therefore you have to use the traditional way to do the eye calibration and then run the experiment.

If you want to test the calibration result, you can turn on the debug mode (Debug 1 '1/0=On/Off') and then run experiment, but you will feel slow motion of experiment.



For Debug Mode:

If you set debug mode in parameter file, you will see many stuff on screen.

- 1) Picture (either circle or square)
- 2) All possible paths (blue lines)
- 3) Nodes in path (red square dot). The image will follow the red dot path moving.
- 4) Mask
- 5) Observation windows (white rectangle lines)
- 6) Mouse position: $(x,y) = (825,543)$ in pixel (the yellow circle dot position)
- 7) Quitting conditions (For detail, please see **Setup the Experiment** – Structures Attribute and Phase Quitting Conditions part)

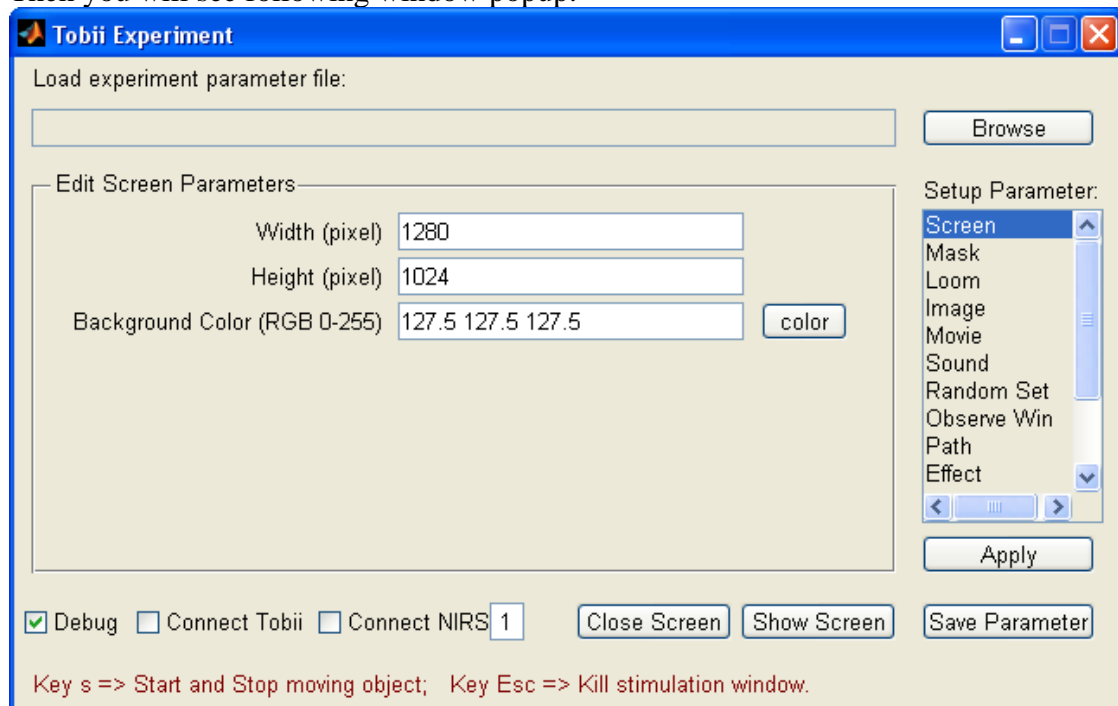
Notice: If **ConnTobii On** in parameter file (ConnTobii 1 '1/0=On/Off'), the yellow dot is the eye gaze position. Therefore you can check that the eye calibration is correct or not.

Setup the Experiment

Our goal is to setup a parameter text file. This file will save all the setting of program and use it in real experiment.

- 1) Create your own folder: You can create a folder for special experiment that includes all the resource such as pictures, sound files and parameter file. After experiment, all data will also save to this folder by default. In this example, I created a folder "TobiiUserData" on Desktop.
- 2) Start MATLAB application: Here we use MATLAB_R2007b version.
- 3) Change the current directory to your created folder:
 - a. You can use the browser button to select your folder, Or
 - b. In command windows: `>> cd /Users/johnny/Desktop/TobiiUserData`
- 4) Start experiment setup program: `>> TobiiGUI`

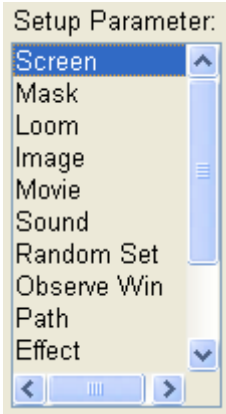
Then you will see following window popup.



- 1) You can select **Setup Parameter** list. Then you will see a different setup panel.
- 2) After you edit parameter, you can click **Apply** button to check any error.
- 3) You can save your parameter text file by click **Save Parameter** button.
- 4) If you have a parameter file, you can load it by click **Browse** button.
- 5) You can run the experiment by click **Show Screen** button.
- 6) If you check the checkbox **Debug**, the stimulation window will draw and show a lot of information to help you design your experiment. Please see **Eye Calibration and Debug Mode** page.
- 7) Since **talk2tobii** mex function doesn't support MATLAB GUI, we cannot use **Connect Tobii** checkbox.
- 8) You can check the **Connect NIRS** with the **serial port number** for output the signal to NIRS machine by serial port.
- 9) The button **Close Screen** will not work. Use key 'Esc' to kill stimulation window.

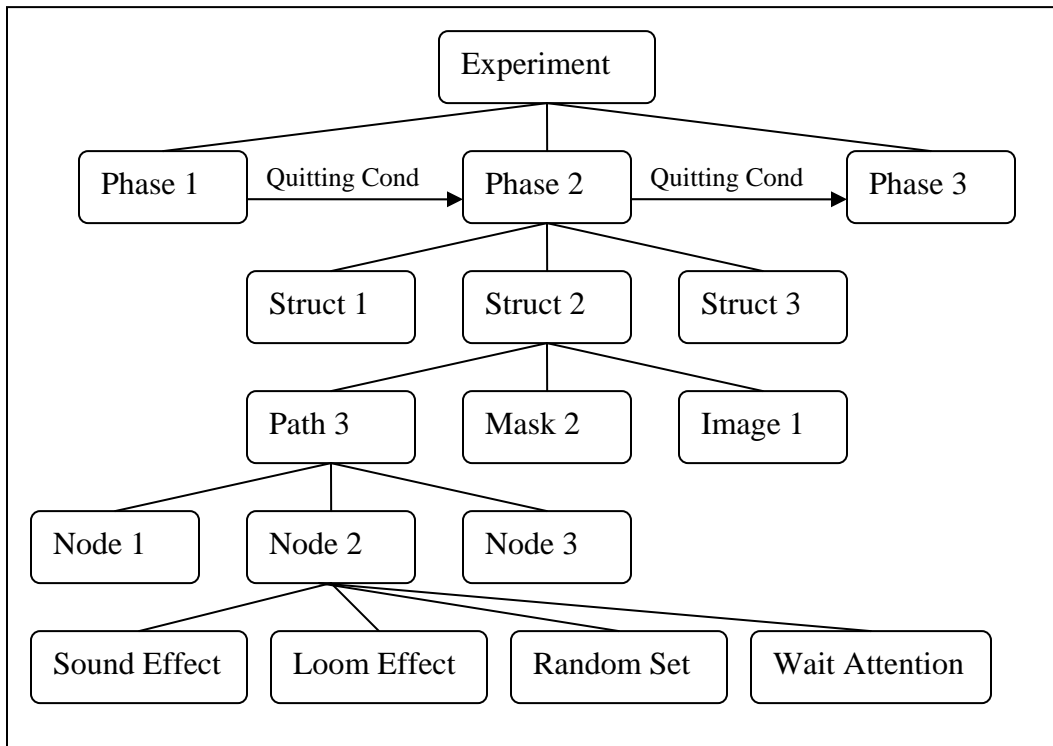
Edit Parameters Detail

There are two main parts of parameter:

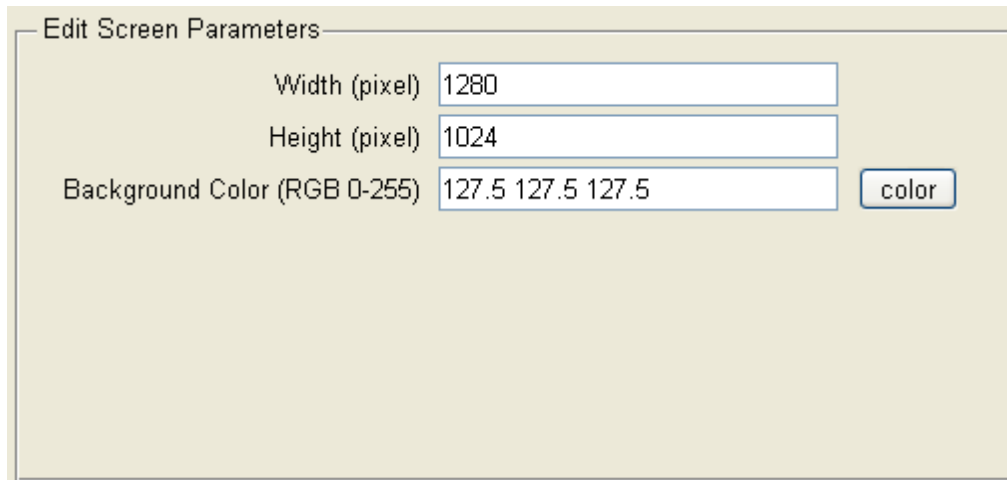


| Resources | Design experiment |
|-------------|-------------------|
| Screen | Structure |
| Mask | Phase |
| Loom | Struct Attribute |
| Image | Quit Phase |
| Movie | |
| Sound | |
| Random Set | |
| Observe Win | |
| Path | |
| Effect | |

Following diagram show the simple experiment logic.

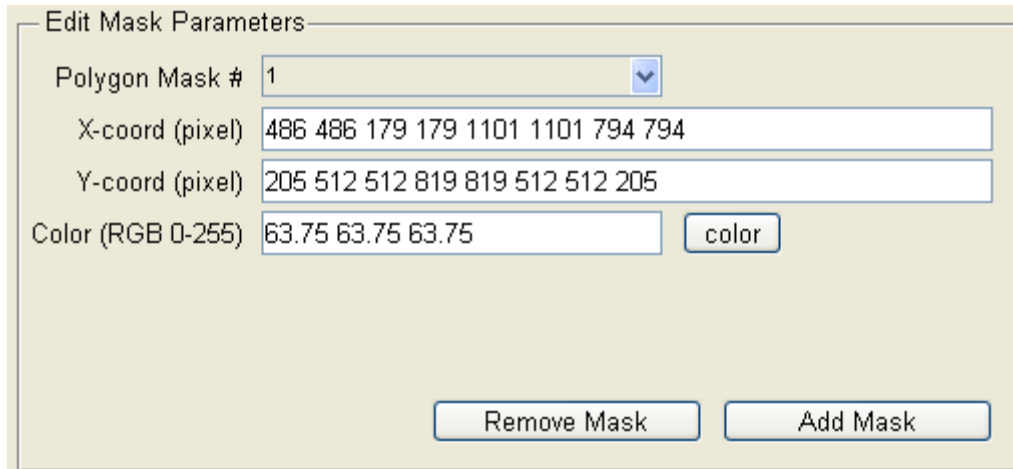


1) Edit Screen Parameters:



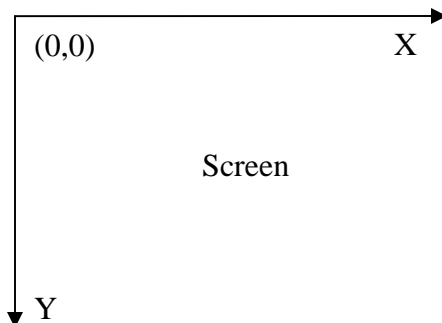
- **Width:** Width of stimulation window in pixel (For Tobii system, width = 1280)
- **Height:** Height of stimulation window in pixel (For Tobii system, height = 1024)
- **Background Color:** You can use the **color** button to select your color or directly input RGB value in range 0-255.

2) Edit Mask Parameters:

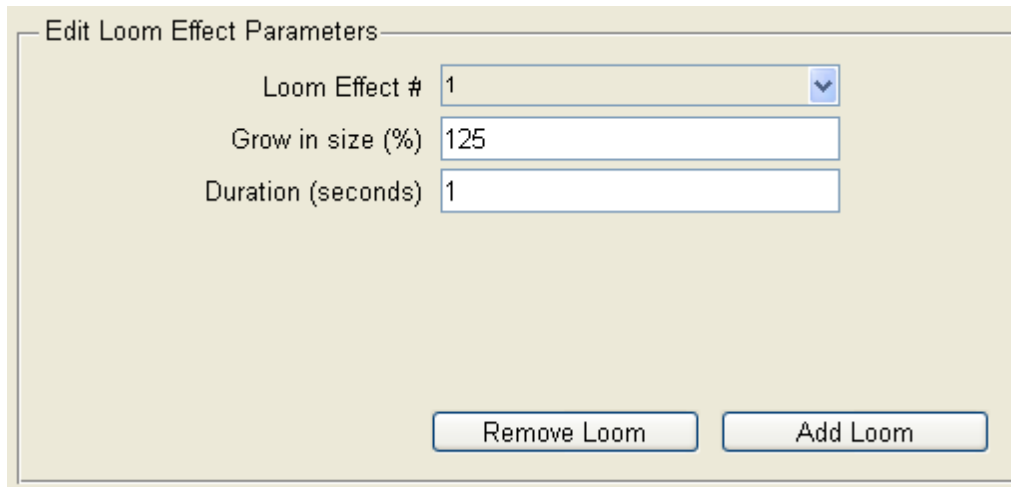


- You can have many polygon masks that need a set of (x,y) positions in orders.
- If you **Add Mask**, program will copy selected mask value to new mask. All other **Add** buttons in different panel has same function.

Notice: The coordinate system is not same as usual. The top left corner is (0, 0) position. This is the coordinate system in **Psychtoolbox**. We always follow this coordinate system in this program.



3) Edit Loom Effect Parameters:



Edit Loom Effect Parameters

Loom Effect # 1

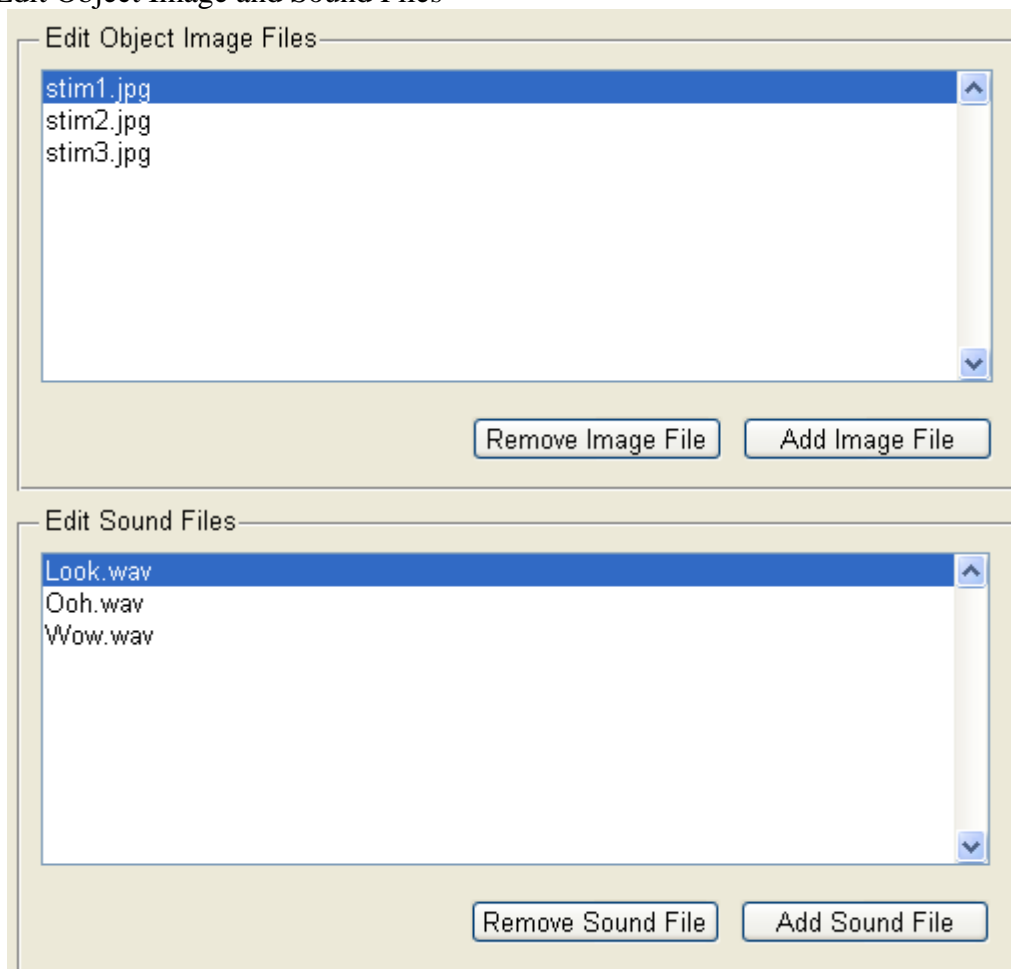
Grow in size (%) 125

Duration (seconds) 1

Remove Loom Add Loom

- Loom effect will grow the size of image from current size (100%) to large (125%) and then back to origin (100%) again within the duration time (1s).
- You can create many different loom effects.

4) Edit Object Image and Sound Files



Edit Object Image Files

stim1.jpg
stim2.jpg
stim3.jpg

Remove Image File Add Image File

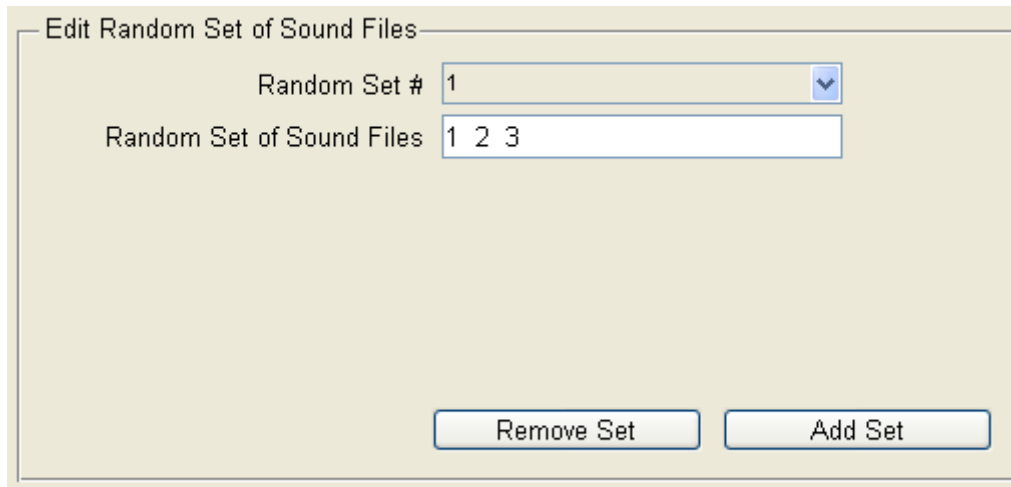
Edit Sound Files

Look.wav
Ooh.wav
Wow.wav

Remove Sound File Add Sound File

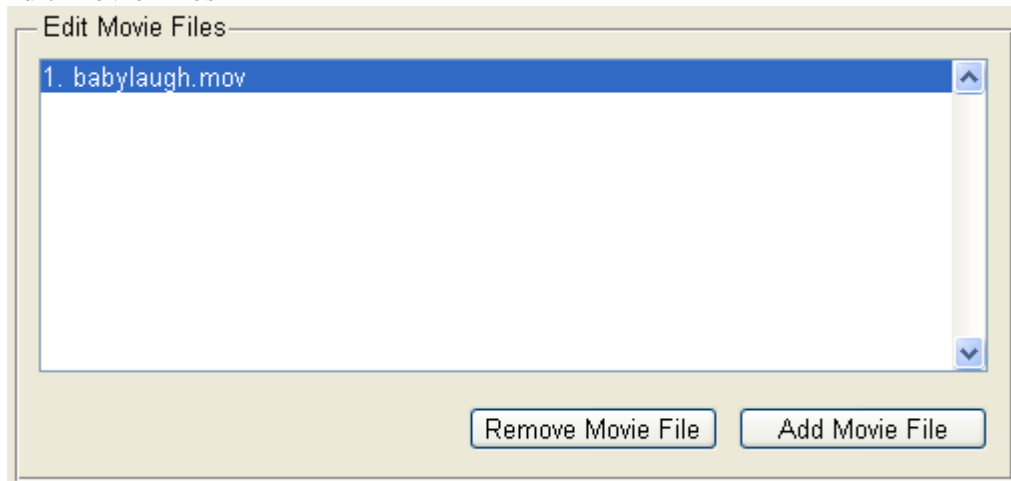
- You can have many different image and sound files.
- Size and dimension of input image are not important. We will crop it in square from the center of image and then map (texture mapping) to desired size.

5) Edit Random Set of Sound Files



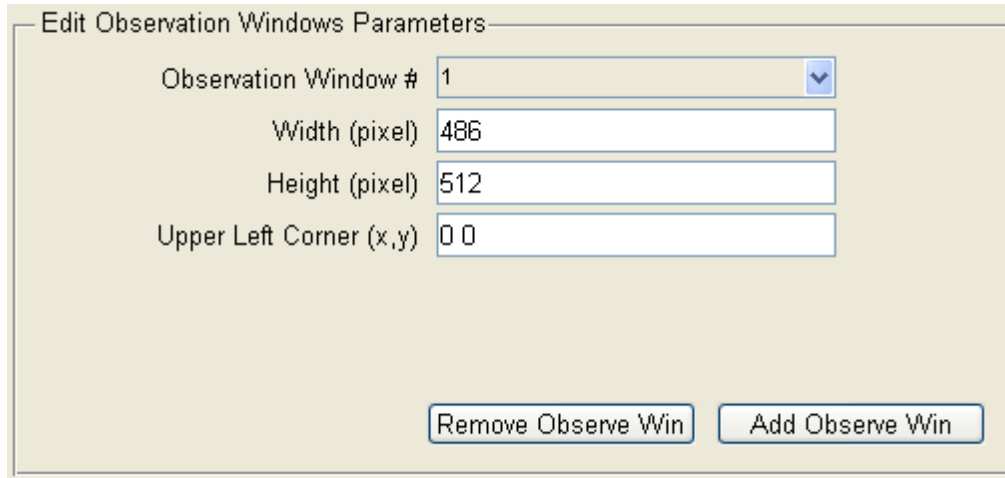
- If you want the experiment that can randomly select sound files, you need a random set of sound files.
- Random Set is an index of sound files, so the number of the set must within the range. For example, if you have 3 image files, then you only can use 1, 2 or 3.
- You can use a random set on any node of any path.
- If you remove the random set, you have to modify the node effect.
- There is no limitation how to arrange the set. Following examples are all correct and different set could have uneven probability for different sound file.
 - 1 2 3
 - 1 2 2
 - 3 1 2 3 1

6) Edit Movie Files



- You can have many different movie files such that media files can be handled by use of Apples Quicktime-7 API, such as .mov file.
- Size and dimension of input movie are not important. We will crop it in square from the center of movie and then map (texture mapping) to desired size.

7) Edit Observation Windows Parameters:



Edit Observation Windows Parameters

Observation Window # 1

Width (pixel) 486

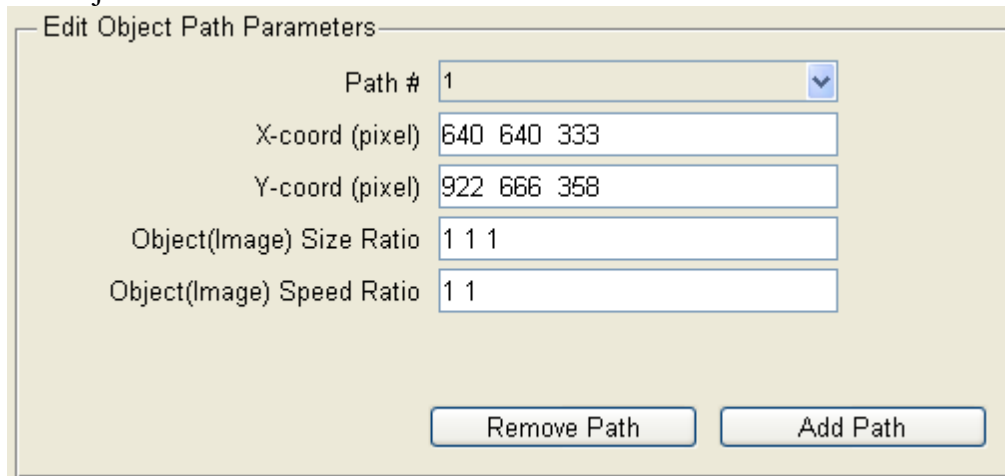
Height (pixel) 512

Upper Left Corner (x,y) 0 0

Remove Observe Win Add Observe Win

- We draw the observation rectangular window by its upper left corner location (x,y), width and height.
 - You can have many different observation windows.
- Notice:** It designs for interactive reaction during the experiment, such as quitting phase.

8) Edit Object Path Parameters



Edit Object Path Parameters

Path # 1

X-coord (pixel) 640 640 333

Y-coord (pixel) 922 666 358

Object(Image) Size Ratio 1 1 1

Object(Image) Speed Ratio 1 1

Remove Path Add Path

- The image object will move along the path in straight line between nodes.
- You need give the locations (x, y) of nodes in order.
- You can set the object (Image) size ratio for each node. The object size will change linearly form node to node. By default, ratio = 1.
- You can set the object (Image) speed ratio for each node. The object moving from node 1 to node 2 use speed ratio at node 1, so the last node do not need speed ratio and should not set it. By default, ratio = 1.
- You can have many different paths.
- You can add different effect on each node of each path.
- If you modify the nodes in path, you have to modify anything relative it, such as Effect or Quitting Phase.

9) Edit Effect on Nodes of Path

Edit Effect on Nodes of Path

Path # 1

Node # 1. (640, 922)

Sound Effect None

Loom Effect None

Sound Random Set # 1

Attention(observation) Win # 1 Wait Attention

Press key 'g' to quit the Wait Attention anytime.

- Now you can set the sound or loom effect on each node of each path.
- Effect will happen when image object reaches the node.
- You can select any sound effect from your input sound file.
- You can select any loom effect that you made it before.
- You also can select random or none effect.
- If you select random sound effect, then program will use the sound random set to pick up the sound file.
- If you check the Wait Attention, the image will stop at the node and repeat sound and/or loom effect (Even if there is no any effect) until baby eye look at Attention Window. Notice: You can press key 'g' to quit the **Wait Attention** anytime, even if baby is not looking at the Attention Window.

10) Construct Structures

Construct Structures

Structure # 1

Path # 1 Mask # 1

Image 1. stim1.jpg

Image: Shape Circle Diameter 164 Speed (pix/s) 200

Use Front Img 1. stim1.jpg Use BG Img 1. stim1.jpg

Use Reward mv 1. babylaugh.mov with Shape Circle Delay(s) 0.5

Movie: center(x,y) 640 512 Width 150 Duration(s) 10

Remove Struct Add Struct

- You can create many structures for your experiment.
- Each structure has one path, one mask and one image.
- Image shape could be circle or square.
- Image size depends on circle diameter (square length).
- The speed of image is pixel/sec.
- If you use Foreground Image, screen will not draw anything else.
- You also can use Background Image instead of the background color.
- You can give the **Reward Movie** at the end of trial. After object (image) move to the last node and disappear, it will wait **Delay** time and start playing movie at **center(x, y)** in **circle or square** within dimension **Width**. Movie will playback with endless loop within **Duration**.

11) Edit Experiment Phase:

- You can have many phases in experiment. They will execute in order until finish all of them.
- Each phase has total trials number and maximum time period. The phase will stop when either one condition is satisfied.
- You can set waiting gap between trails. (Wait between (s) = 1)
- **Struct Weight** is useful when you check the **Random** checkbox.
- Uncheck **Random**: experiment will follow the **Struct# in order** and repeat them until finish all trials.
- Check **Random**: experiment will randomly pick up one from the **Struct# in order**.

For example,

Struct# in order = 1 1 1 2 2

Around 60% of Struct# 1 and 40% of Struct# 2 (Not exactly!) will be selected.

- Uncheck **Fix**: experiment will not care about the **Struct Weight**.
- Check **Fix**: experiment will make sure the **Struct Weight** when randomly pick up structure.

For example,

Struct# in order = 1 2

Struct Weight = 7 3 (70% 30%)

Total trials # = 10

We will still randomly pick up from Struct# 1 or 2, but we will make sure the outcome in Struct# 1 is 7 and Struct#2 is 3 for total 10 trials.

- You can set mask opaque percentage.

For example,

Start% = 50 Inc Step% = 10 End% = 100

Trail # 1: Opaque 50%

Trail # 2: Opaque 60%

Trail # 3: Opaque 70%

Trail # 4: Opaque 80%

Trail # 5: Opaque 90%

Trail # 6: Opaque 100%

Trail # 7: Opaque 100%

...

- You can skip the phase and go to next phase by pressing the “p” key. Then after current trail finished, experiment will start next phase first trial.

12) Edit Structures Attribute (Define good trial)

- Structure Attribute defines a good trial or bad trial. A complete trial means the image finish whole path.
- You only will define the structure attribute when you use the **Quitting Condition** in Phase, because we only use good trial to calculate the quitting condition.
- **Interesting Node** defines start node and end node. We only use the data between them for calculation any conditions.
- For each structure, you can define which observation windows are correct or incorrect. The windows# can be more than one, but they have to be defined before. The number in the bracket (1-3) remind you which number can use.
- **Above Wins: Require Looking%** means that

$$\text{Require Looking\%} \leq \frac{\text{Baby looking at the correct or incorrect obsWins}}{\text{Baby looking at whole screen}} \times 100\%$$
- **Look at screen time** require baby look at screen time during the trial. There is the total time from start node to end node in the bracket (e.g. 3453ms) for reference. Notice that the time doesn't include any loom effect time at nodes.
- **Validity Level** indicates the data reliability. 0 is Certainly and 4 is Certainly not. If the data is not reliable, we will not use it.

Example of good/bad trail calculation in real program,

- 1) Suppose we are interested on data that from node 1 to 3 and the total number of frames is 206 between node 1 to 3. Then we will collect one gaze point data of eye for every frame, so we have 206 gaze data points.
- 2) Suppose we set the Validity is 2 and there is 155 gaze data points with the Validity smaller or equal to 2.
- 3) Suppose there are 150 gaze data points inside the screen. (It is possible that we have some data is a little bit outside the screen and Validity is OK.)
- 4) Within those 150 gaze data points, we have 75 gaze data points that baby looks at the any correct or incorrect observation windows. That means baby looks at ObsWins $75/150 = 50\%$.
- 5) Suppose the monitor has 60Hz refresh rate. Then baby looks at whole screen time is $(150 / 60) \times 1000 = 2500\text{ms}$.
- 6) If both 50% looking at ObsWins and 2500ms looking at whole screen fulfill requirements, then it is a good trail. If not, it is a bad trail and will not use for calculating in quitting condition.

13) Edit Phase Quitting Conditions

- If you don't check the **Use Condition**, the program relies on the **Total trials#** and **Max time** to quit the phase.
- If you check the **Use Condition**, then the program will use the quitting condition to quit the phase. That means the phase will quit until it fulfills all conditions whatever use how many trials or time.
- If you check **Use Total trials** checkbox, it will quit anyway when reach the **Total trails #** in Phase panel.
- **Interesting Structs #**: we only use interesting structure for calculating conditions. The number in bracket (1 2) remind you what structure can use.
- **Last n trails**: We will use last n trial to calculate conditions. Those trials must be an **Interesting Structures** and **Good Trail** that defined in **Structure Attribute**.

- **Total Looking:**

All calculation within **Interesting Nodes**.

Total looking time at **Correct ObsWins** of each trial

Let $R_i = \frac{\text{Total looking time at Correct ObsWins of each trial}}{\text{Total looking time at Corr/InCorr ObsWins of each trial}}$

Total looking time at **Corr/InCorr ObsWins** of each trial

Where $i = 1, \dots, n$ and n is last n trials.

Correct Look% $\geq \frac{\sum_i^n R_i}{n}$, where n is last n trials.

- **First Looking:**

It is a correct first looking, if baby first looks at **Correct ObsWins** in the **Interesting Node**. Here **Interesting Node** and **Correct ObsWins** define in **Structures Attribute**.

Correct Look# \geq Total number of correct first looking of last n trial

- **Ttest**: Matlab function

$[h, p, ci] = \text{ttest}(\dots)$ returns a $100 \cdot (1 - \alpha)\%$ confidence interval on the population mean, or on the difference of population means for a paired test. Suppose we collect Correct Look% of each last n trials and put them in an array. For example,

CorrLookArray = [40% 50% 70% 60% 80%]

$[h, p] = \text{ttest}(\text{CorrLookArray}, \text{mean}\%)$

If $p \leq \alpha$, it passes the ttest and with $100 \cdot (1 - \alpha)\%$ confidence.

- **Logic:**

You have to select one condition in first drop down list and you can select all none on other drop down list. Notice that we will always do the logic inside the bracket first. Following is some possible logic examples.

| | | | | |
|----------------|------|----------------|------|-------|
| (Total Looking | None | None) | None | None |
| (First Looking | And | Total Looking) | None | None |
| (Total Looking | Or | First Looking) | And | Ttest |
| (First Looking | None | None) | And | Ttest |

- If any condition (Total Looking, First Looking or Ttest) passed before, they **always pass** in future trails and just wait for other condition to pass.

Example,

According to setting of above figure, we have following example.

(Total Looking Or First Looking) And Ttest

Let Tn be the trail number.

Let Fn be the total number of frame between **start node and end node**.

Let Fv be the frames with good **Validity** and inside the screen.

Let Fc be the frames that gaze point in **correct** observation windows.

Let Fi be the frames that gaze point in **incorrect** observation windows.

Let F1 be the correct first look at correct observation windows.

Let Tg be a good trail.

| Tn | Fn | Fv | Fc | Fi | F1 | Tg |
|----|-----|-----|-----|----|----|----|
| 1 | 200 | 170 | 100 | 50 | 1 | 1 |
| 2 | 200 | 10 | 1 | 2 | 1 | 0 |
| 3 | 200 | 190 | 110 | 20 | 0 | 1 |
| 4 | 200 | 180 | 30 | 70 | 1 | 1 |
| 5 | 200 | 185 | 80 | 70 | 1 | 1 |
| 6 | 200 | 160 | 110 | 20 | 1 | 1 |

$$\text{Total Looking} = \frac{\left(\frac{100}{100+50} + \frac{110}{110+20} + \frac{30}{30+70} + \frac{80}{80+70} + \frac{110}{110+20} \right)}{5} \times 100\%$$

Total Looking = 63.846% > 60%, so it pass the total looking condition.

First Looking: Correct Look# = 4 > 3, so it pass the first looking condition.

Ttest: in matlab command window

```
>> CorrLookArray = [100/(100+50) 110/(110+20) 30/(30+70) 80/(80+70) 110/(110+20)];
```

```
>> mean = 50/100;
```

```
>> alpha = 0.05;
```

```
>> [h, p] = ttest(CorrLookArray, mean, alpha, 'right')
```

Result: h = 0 and p = 0.12524811832192 (which means only have 87.5% confidence above the mean.) It is false for ttest.

Logic: (Total Looking Or First Looking) And Ttest → False

That means it will not quitting the phase, until it fulfills all conditions.

In the debug mode of program, you are able to test the logic is correct or not. There is some information will show on the screen and matlab command window. If you don't check the Connect Tobii, you can use the mouse instead of eye to test the program.

Notice: When you see the information on screen, it tells you last trial information.

Hits: Usually the program will slow down in the debug mode. You can increase the **Speed** of image for testing purpose.

For example,

```
P#=1, S=1, I=1, LookScr=3533ms, ObWs/Whole=0.642, Good=1|1, FirstLook=2|0, CorrObWs%=0.493
P#=1, S=2, I=1, LookScr=2233ms, ObWs/Whole=0.537, Good=1|2, FirstLook=2|1, CorrObWs%=0.667
P#=1, S=1, I=1, LookScr=3533ms, ObWs/Whole=0.500, Good=1|3, FirstLook=1|1, CorrObWs%=0.745
P#=1, S=2, I=1, LookScr=2233ms, ObWs/Whole=0.500, Good=1|4, FirstLook=1|0, CorrObWs%=0.746
P#=1, S=1, I=1, LookScr=3533ms, ObWs/Whole=0.486, Good=1|5, FirstLook=1|1, CorrObWs%=0.728
Cond: TL=0.659|1, FL=3|1, TTest=0.021594|1 Next=1
P#=2, S=1, I=1, LookScr=3533ms, ObWs/Whole=0.382, Good=1|1, FirstLook=1|1, CorrObWs%=0.827
```

- **P#=1:** Phase number is 1.
 - **S=1:** We use structure 1.
 - **I=1:** If I=1, then it is interesting structure. If I=0, it is not.
 - **LookScr=3533ms:** Looking at screen time during the trail within the interesting nodes.
 - **ObWs/Whole=0.354:** Looking at ObsWins time / Looking at whole screen
 - **Good=1|2:** There are two parts. First part 1/0 means it is a good/bad trial. Second part 2 means how many good trials we accumulated right now.
 - **FirstLook=2|1:** First part 2 means which ObsWin# is first looking by baby. If baby don't look at any ObsWins, then you will see number 0. Second part 1/0 means it is a correct/incorrect first looking.
 - **CorrObWs%=0.672:** Correct ObsWins time / All ObsWins time.
- Cond:**
- **TL=0.659|1:** First part 0.659 means Total Looking condition final percentage after last n good trials. Second part 1/0 means it pass/fail the requirement.
 - **FL=3|1:** First part 3 means how many first look of last n good trial. Second part 1/0 means it pass/fail the requirement.
 - **TTest=0.021594|1:** First part 0.021594 is p value of ttest. Second part 1/0 means it pass/fail the requirement.
 - **Next=1:** 1/0 means we should go to next phase or not after check all conditions.

Screen will show some information when it is available. Matlab command window will print out the information, so you can check it after quit experiment.

Data Analysis

There are two Matlab functions for general data analysis. You can run these two functions in either command window (Terminal or xterm) or Matlab application.

1) **CreateCsvData**(inputFilename, outputFilename)

This function will create an output .csv file according to input file (experimentData.mat by default) and a column title .txt file.

In command window

- i) change current directory to your created resource folder
`>> cd /Users/aslinlab/Desktop/TobiiUserData`
- ii) run CreateCsvData
`>> CreateCsvData ('experimentData.mat', 'CsvData')`

Then you will find two new files in your folder.

- 1) CsvData.csv – pure data file that can be opened by Excel.
- 2) CsvData.txt – column title file that show what kind of data in .csv file and phase result.

Columns in CsvData.txt:

TobiiTime(ms), CallTime(ms), ResponseTime(ms), Phase#, Trial#, Path#, Node#, LeyeValidity, ReyeValidity, gazeX, gazeY, waitAttention, ObjCenterX, ObjCenterY, ObjSize, Obswin1, Obswin2,... ObswinN, ElsewhereWin

You can modify the “Obswin#” name, such as LeftWin or RightWin that will be used for label on the figure in the future.

Data explanation:

- After experiment start, we will record the data on each frame. Tobii monitor is 60 Hz, so we get 60 data per sec. Each row in .csv file is one set data.
- Column **TobiiTime(ms)**: Tobii gets sample (gazeData) time that is 50 Hz.
- Column **CallTime(ms) and ResponseTime(ms)**:: This is the Mac computer clock time right **before** and **after** we call the TALK2TOBII('GET_SAMPLE') to get gazeData. (start in version 3)
- Column **Phase#, Trial#, Path#**: This is experiment information.
- Column **Node#**: This information shows you the moving object location.
 - If the object move from node **N** to node **N+1**, we will mark **N**.
 - If the object doesn't show up on the screen, we will mark 0. If we play movie, we will mark -1. (Node 0 or -1 means no object on screen.)
- Column **LeyeValidity, ReyeValidity**: (0-4) where 0 – Certainly (>99%) and 4 – Certainly not (0%)
- Column **gazeX, gazeY**: gaze coordinate of eye (x,y) in pixel.
- Column **waitAttention**: if you use waitAttention on any node of path, it will mark 1, else will mark 0. (start in version 2)
- Column **ObjCenterX, ObjCenterY**: current object location(x,y) in pixel. -1 means no object on the screen.
- Column **ObjSize**: current object size in pixel. -1 means no object on the screen.
- Column **Obswin1, Obswin2, ... ObswinN, ElsewhereWin**: If eye position in any of observation windows that will be mark 1. If not, it will mark 0. If eye position is not in any of observation windows and it is still on screen, then it will be mark in ElsewhereWin.

Phase result in CsvData.txt:

- 1) Each line shows one phase result.
- 2) Format: Keyword=Value,Keyword=Value,...
so the delimiter will be comma(,) and equal sign(=) . For example,
Phase#=1,TotalTrial#=6,TotalTime(s)=27.100,FinalTrial#=-6,UseQuittingCond=1,Quitting=TotalTrial,..

| Keyword | Value |
|---|--|
| 1) Phase# | Integer 1,2,3.... |
| 2) TotalTrial# | Total trial number that you set for this phase |
| 3) TotalTime(s) | Total time use for this phase. |
| 4) FinalTrial# | Actual total trial number in this phase in the experiment. |
| 5) UseQuittingCond | 1/0 = T/F |
| 6) Quitting | Either TotalTrial or Condition |
| Following information show up when you select UseQuittingCond = 1 | |
| 7) Condition | e.g. (TotalLook Or Ttest) And FirstLook |
| 8) GoodTrial# | Number of Good Trial for interested structures |
| 9) BadTrial# | Number of Bad Trial for interested structures |
| 10) SuccessTL | 1/0 = T/F, Success of Total Look condition |
| 11) TLTrial# | How many good trials used for passing the condition |
| 12) TotalLook | Total Look value when it pass the condition |
| 13) SuccessFL | 1/0 = T/F, Success of First Total Look condition |
| 14) FLTrial# | How many good trials used for passing the condition |
| 15) FirstLook | First Look value when it pass the condition |
| 16) SuccessTtest | 1/0 = T/F, Success of Ttest condition |
| 17) TTTrial# | How many good trials used for passing the condition |
| 18) Ttest | Ttest value when it pass the condition |
| 19) TTPval | Required Ttest p value for passing the condition |

Notice: Any value is -1 that means the value is not available. For example, there are not enough last n good trials, but it goes to next phase because of reached TotalTrial# condition, so we may not find the value of TotalLook, FirstLook or Ttest.

Version:

By default, the function use version 4 that include all columns' data.

Version 1 doesn't include the waitAttention, CallTime(ms), ResponseTime(ms), ObjCenterX, ObjCenterY, ObjSize

CreateCsvData will tell you which version of data. Then you need input this version number in **ObsWinAvgFigure** function.

For example, if you have a data.mat data file

For version 1:

```
>> CreateCsvData('data .mat', 'data ')  
Tobii experiment version: 1  
>> ObsWinAvgFigure('data.csv','data.txt',-1,-1,-1,1)
```

For version 2:

```
>> CreateCsvData('data .mat', 'data ')  
Tobii experiment version: 2  
>> ObsWinAvgFigure('data.csv','data.txt',-1,-1,-1,2)
```

....

For version 4: (default)

```
>> CreateCsvData('data .mat', 'data ')  
Tobii experiment version: 4  
>> ObsWinAvgFigure('data.csv','data.txt')
```

2) **ObsWinAvgFigure**(inputDataFile,inputDataTitleFile,firstNode,lastNode,phaseNum,version)

This function will show a figure

In command window

i) change current directory to your created resource folder

```
>> cd /Users/aslinlab/Desktop/TobiiUserData
```

ii) run ObsWinAvgFigure

```
>> ObsWinAvgFigure ('CsvData.csv', 'CsvData.txt')
```

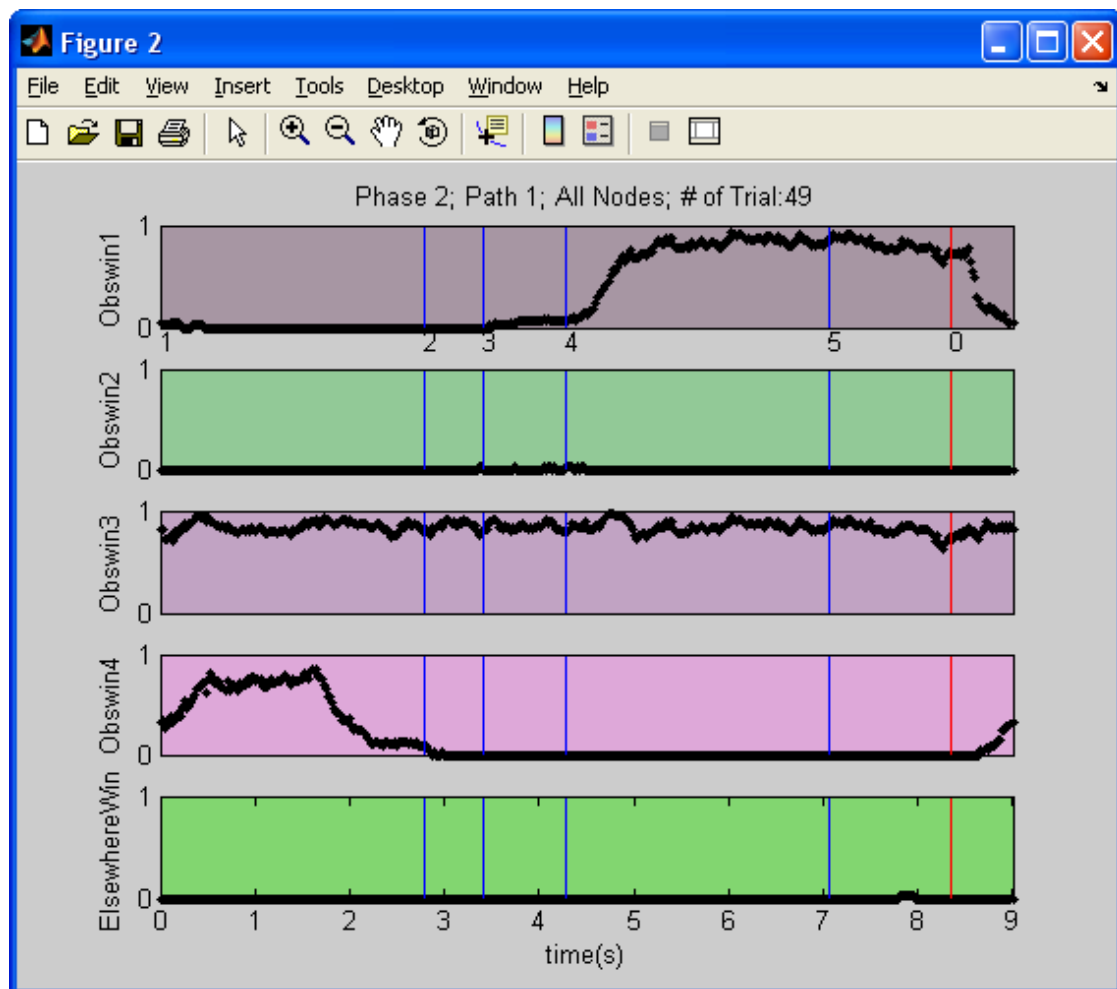
In this case, they will show all the nodes' information in figure.

You also can specify the first Node and last Node, or which phase you are interested.

e.g. >> ObsWinAvgFigure ('CsvData.csv', 'CsvData.txt', 2, 7, 2);

By Default, we use version 3 that match the columns in .csv data. If you create .csv data by version 1 or 2, you have to use same version in **ObsWinAvgFigure** function.

Notice: We will eliminate all waitAttention data, so that we have same number of data on each path with different trials.



This figure shows the result of experiment in Phase 2 and Path 1 for all nodes and total number of trials. Node#1 is always at 0 sec, so we don't draw the line. Then each vertical line shows next node position with node number label. Here we have 5 nodes, but you find the 6th vertical line. That last vertical line indicates the trial finished and there is no object on screen.

Explanation:

First of all, let us find average eye position in left window at Node#5.

Conditions:

- 1) total 100 trials in Phase# 2
- 2) 70 trials use Path# 1
- 3) Path 1 will use 7.2 sec or total 432 frames (7.2x60Hz) to finish the trial.
- 4) Node#5 at 5.7 sec or frame#342

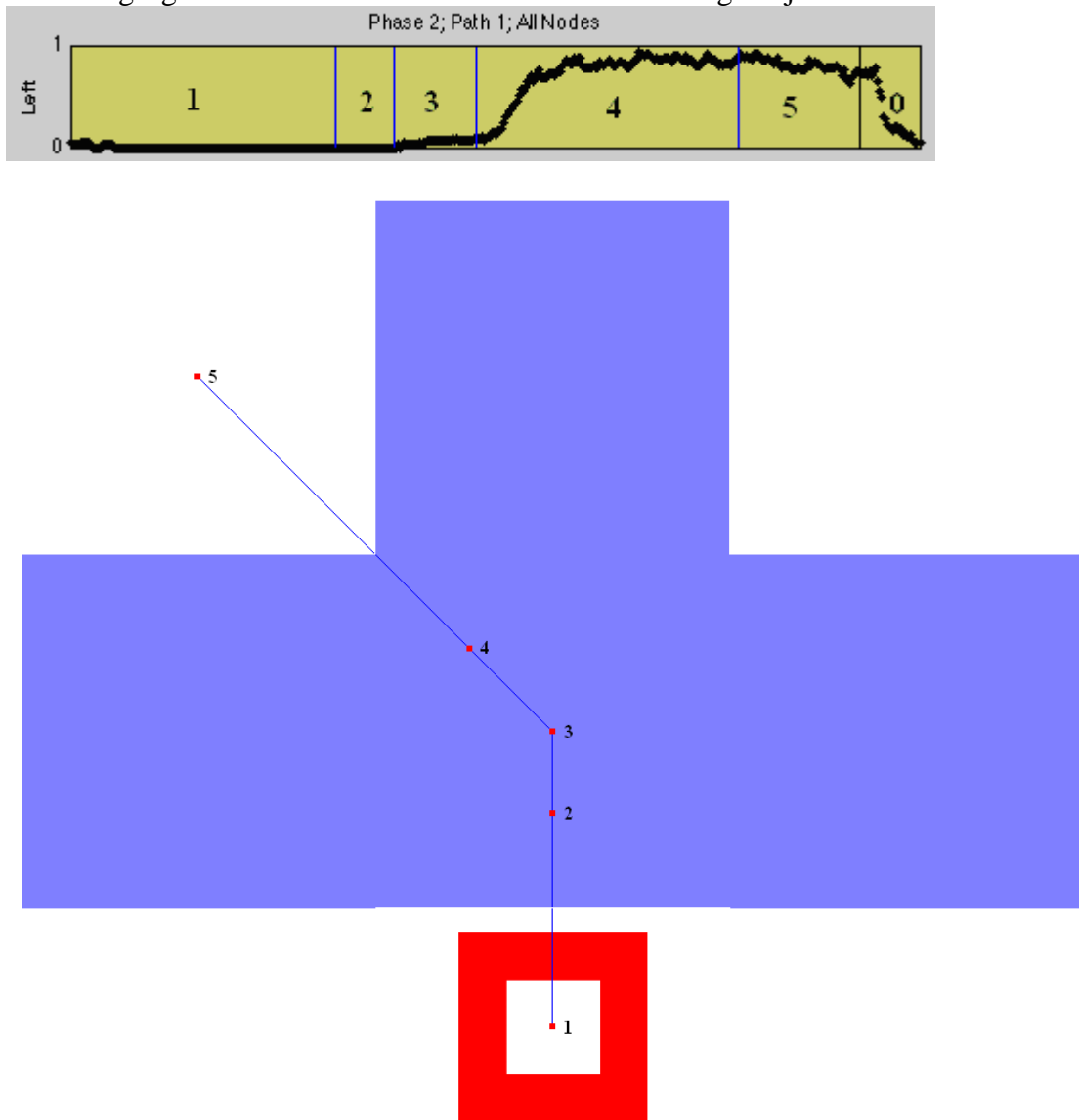
Calculate average eye position in left window:

- 1) When the image object is at Node#5, we will check the eye position in left window or not. If it is in, we will mark 1. If not, we mark 0.
- 2) Suppose we got 6 mark 1 at Node#5 in left windows for 7 trials, then average will be $6/7$ (~86%) in left window at Node#5.

According to above calculation, we also can find any average eye position in particular frame# for any observation windows.

Notice: If the experiment stops in the middle, the last trial data still will be in .csv data file, but it will not use to calculate the average eye position in observation window figures.

Following figure shows how we mark the Node# for image object location.



- 1) Mark Node#1: Object is at Node#1 and moving to Node#2. At Node#1, it use 1 sec for Loom effect and use 1.2 sec to move to Node#2. Therefore, from 0 to 2.2 sec is mark Node#1.
- 2) Mark Node#2: Object is at Node#2 and moving to Node#3. At Node#2, the object is just completely covered by the mask. It uses 0.5 sec.
- 3) Mark Node#3: Object is at Node#3 and moving to Node#4. It uses 0.7 sec. Before the object reached Node#4, the object is still completely covered by the mask.
- 4) Mark Node#4: Object is at Node#4 and moving to Node#5. It uses 2.3 sec.
- 5) Mark Node#5: Object is at Node#5. It uses 1 sec for Loom effect.
- 6) Mark Node#0: This is the time between trials to trial. There is no object image, so we mark 0. It use 0.5 sec.

The total time is around 7.2 sec