**FlyFly User Manual**

**Installing and running FlyFly**

***Current version: 4.0 - Multiple Screen Edition***

***Used for open loop experiments on the tethered flight arena***

[**INSTRUCTIONS FOR INSTALLING FLYFLY 3**](#_l06enc3pezpp)

[INSTALLING FLYFLY ON WINDOWS 4](#_lynds58e5zfm)

[1. INSTALL MATLAB 4](#_ubcngjplrscl)

[2. INSTALL SUBVERSION 4](#_lobjaxfw462n)

[3. INSTALL PSYCHTOOLBOX 5](#_sjki9h83atct)

[4. INSTALL FLYFLY 5](#_d640gyqtby32)

[INSTALLING FLYFLY ON MAC OSX 5](#_1tmtqzcg4elm)

[**Setting up a Webcam for Recording [Jaxon to fill] 7**](#_k47d8z5yvjdx)

[Required Software 7](#_7w607o36deqk)

[FLIR Blackfly (REQUIRES UBUNTU 20.04 OR 22.04) 7](#_kj7of2rklqq9)

[PS3 Webcam (Legacy) 7](#_fj1tezd3bxvc)

[**Operating Flyfly 8**](#_f2xnplio8p17)

[MAIN WINDOW 8](#_fuh3mehkornl)

[SETTINGS WINDOW 9](#_29oky4nhclja)

[STIMULUS WINDOW 11](#_za2rw3spzo1)

[1. Stimulus Parameters 12](#_8bhf14tzw6br)

[2. Layer Settings 13](#_jnv0axi2g4oh)

[3. Layer Controls 14](#_95z9cpwma6qe)

[4. Trial Controls / Sequencer 17](#_gr72nri2vuus)

[5. Execution Controls 17](#_lo00tsqilaz1)

[6. Stimulus Navigation Controls 18](#_54cjnicryxx)

[7. “Interpolate” / Editing Controls 18](#_z03iw86n4f4k)

[8. Screen Info 19](#_dg02cpcsrqhu)

[9. Impulse checkbox 19](#_p3zrbcxyr0el)

[10. Stimulus and Layer Name Controls 20](#_szj37tgjb46b)

[11. “File” menu 20](#_akhl7m1nvwpp)

[12. “Tools” menu 20](#_26w2m12xr7cg)

[13. “Special” menu 20](#_rwvl9hen6c00)

[14. M-sequence controls 20](#_ctg37l1xml2o)

[15. Gaussian White Noise controls 21](#_4znvn3mb9aty)

[16. Paloma XY generation 21](#_waatkr6qgyb4)

[17. High Definition Image 21](#_1itx57yqz7ty)

[Version 3.2 change 21](#_8evvtex5k24h)

[Version 3.3 Change 22](#_wo0ohmu5pueo)

[**MAIN WINDOW: STIMULI 23**](#_pq4khprs0pys)

[RECT TARGET 24](#_h19nouqk0y6h)

[**IMAGE TARGET (NB - Sarah needs to check/fix highlighted parts) 25**](#_w8uqitvack0v)

[3D TARGET 27](#_eecrnu4hzoex)

[SINE GRATING 29](#_wdbo4z2ag4h6)

[COLOR FILL 30](#_u5c3zwkijkss)

[ROLLING IMAGE 31](#_7ln46o7jrywo)

[ROLLING IMAGE MII 33](#_kjkbup9aegut)

[STARFIELD 2: 3D Space 35](#_tlvo1soaop5l)

[STARFIELD 3: JUMPS 38](#_cwfq6q3biqoq)

[SINE GRATING RF 39](#_g8ia4s9z5nha)

[APERTURE 40](#_q5ongflscchf)

[DUAL APERTURES to be completed by Sarah 41](#_4r7cte8jpcwp)

[FLICKER RECT 42](#_4gusgzigmnr1)

[MOUSE TARGET 43](#_sd9rvhsxbyho)

[GRID 44](#_jzwt9qirb5xq)

[.MAT SEQUENCE 45](#_r5vq8qtyk7qy)

[PALOMA TARGET REPLICATION 46](#_ttsk42bkwvmw)

[LOOM 47](#_i144xlgg9ew7)

[WHITE NOISE 49](#_njfsdps1losd)

[TEXT STRING 50](#_nh1hwatc7f2h)

[Parameter saving structure 51](#_zb8spdyotygh)

[stimulus 51](#_2ogxm8b6m4ht)

[debugData 51](#_x8qlbuwvzpmj)

[Note on the interpretation of timeStart 52](#_y10h3snxvy66)

# INSTRUCTIONS FOR INSTALLING FLYFLY

**Note**:

FlyFly uses the software package PsychToolbox to present stimuli on-screen and to manage precise timing of screen frames. The makers of PsychToolbox strongly advise against using their software on Mac OSX, and seem to be only luke-warm in endorsing the use of Windows. They strongly recommend running PsychToolbox on Ubuntu, and consequently we use Ubuntu for all our stimulus computer installations of FlyFly.

This manual is for installing FlyFly on a computer that is NOT a stimulus computer**,** i.e. it will NOT be used for presenting stimuli during experiments. Examples of non-stimuli computer usage may be that you want to run FlyFly simply to see what the stimuli look like, or because you want to do software development on FlyFly itself. **If you want to install FlyFly on a stimulus computer** (which should therefore be dual-boot Ubuntu with some other operating system), **please instead consult the document “User manual for setting up a stimulus computer for running FlyFly at high temporal resolution”** (on the Google Drive as “InstallationManualFeb2024.pdf”).

If you want to install FlyFly on a **non-stimulus computer** on

**Ubuntu**: follow steps 4 and 5 in the above-mentioned “User manual for setting up a stimulus computer for running FlyFly at high temporal resolution” (“InstallationManualFeb2024.pdf”). (Steps 1, 2 and 3 relate to configuring a graphics card and screens for precise timing, and so can normally be skipped for non-stimulus computers, although it can’t do harm if you follow them as well).

**Other flavours of Linux**: we have not used FlyFly under other Linuxes, but the Ubuntu instructions would probably be the best place to start. Whilst untested, best results are likely found with other debian based OS, if you are intending to use a different distribution.

**Windows**: See “Installing FlyFly on Windows”.

**Mac OSX**: See “Installing FlyFly on Mac OSX”.

## INSTALLING FLYFLY ON WINDOWS

Note: **You must be logged in as an administrator** in order to install the software for all users. You must install and run all programs as the administrator.

Flinders University-specific instructions:

* There are two kinds of Flinders computers: ones with the “standard” image, and ones with the much less restrictive “lab” image. We have experience of installing FlyFly on Macs with the standard image, and PCs with the lab image, but not yet for the other two possible combinations.
* If the computer is a uni machine with the special “lab” image, the following will work. BUT keep in mind that all software should be installed in either the “Program Files” folder, or under “C:\Applications” (we will be prevented from installing software elsewhere).

### 1. INSTALL MATLAB

Follow normal procedure to install MATLAB.

* FlyFly will work with MATLAB 2016a and beyond on Windows
* FlyFly is recommended to be run on MATLAB 2023b, as this is what the compatibility changes were made on and should hence have the best performance with
* For setups requiring the filming of the participant, ensure the official toolboxes ‘Image Processing’ and ‘Image Acquisition’ are installed alongside your version of MATLAB
* Flinders Uni specific:
  + We have done installation by logging in to a MathWorks account (of any lab member) that is already connected to the uni’s Headcount licence, and downloading, installing and registering MATLAB via that account.
  + The installation process needs to be done by an administrator, after which any Flinders user of that machine will be able to use MATLAB.
  + So far, the person logging in to the machine as administrator has also been the person who has logged in to MathWorks (and so linked the uni’s licence to their own identity). It may not be necessary for them to be the same person, but this is untested.
  + Remember to install into one of the two folders (“Program Files” or “C:\Applications”) set aside for software installation.

### 

### 2. INSTALL SUBVERSION

* + Unfortunately, subversion functionality has been removed from Github as of Jan 8th, 2024. This means that the old installation method of PsychToolbox-3 no longer works. See updated PsychToolbox installation instructions below.

### 3. INSTALL PSYCHTOOLBOX

* The best approach is to visit the Psychtoolbox website for the latest download, as well as installation instructions (<http://psychtoolbox.org/>).
* To install PsychToolbox, you first need to first connect to the NeuroDebian repository, which houses the most up-to-date and stable version of PsychToolbox. To do this, navigate to <https://neuro.debian.net/> and select both your operating system (If you are unsure what this is, open up a terminal and enter ‘**lsb\_release -a**’ to find your Ubuntu release) and a server to download from (Recommended to choose either Chinese server for Australian users).
* Click on ‘all software’ when the option appears and you should be provided with two terminal prompts to copy & paste into your terminal. Do this, and you will be connected to the NeuroDebian repository.
* Once this is done, run the commands ‘**sudo apt-get update**’ and then ‘**sudo apt-get install matlab-psychtoolbox-3**’ to install the toolbox to your Ubuntu system
* To install Psychtoolbox-3 to your MATLAB installation, first run the command ‘**ptb3-matlab**’ in a Linux terminal. This will open up a MATLAB instance with Psychtoolbox-3 active. To permanently add the toolbox to your MATLAB installation, first write ‘**savepath**’ into the MATLAB command window. This will save the toolbox and ensure Psychtoolbox is active whenever you open MATLAB next. Then, run ‘**PsychLinuxConfiguration**’ in the command window to begin the configuration process to set up all the files needed to run the toolbox with Flyfly. Answer yes to any user questions that pop up until the configuration script is complete, at which point Psychtoolbox-3 is setup for use with Flyfly. To verify that you have done this correctly, restart MATLAB and run ‘**PsychtoolboxVersion**’ in the command window. The installation was done correctly if a version greater than 3.08 is displayed in response.

### 4. INSTALL FLYFLY

* Download the latest version from <https://github.com/HoverflyLab/FlyFly>, and unzip the file into an appropriate location (which may as well be under C:\Applications in the case of a Flinders lab-imaged machine).
* Set the active directory in MATLAB to the top FlyFly folder, and test the installation by entering

flyfly

at the MATLAB prompt.

## **INSTALLING FLYFLY ON MAC OSX**

Need something here!

# Setting up a Webcam for Recording [Jaxon to fill]

**Relevant for the rotated video version of Flyfly only**

## Required Software

## **FLIR Blackfly (REQUIRES UBUNTU 20.04 OR 22.04**)

To get the FLIR cam successfully running on your computer, first connect it via its unique usb3 adaptor, ensuring the cord is fully connected using the two screws at the end of the cable. Set aside and go to the following website to get the necessary drivers for the camera to run; <https://www.flir.com.au/products/spinnaker-sdk/?vertical=machine+vision&segment=iis>

Once downloaded for linux, extract the files into an appropriate folder and open a terminal in the newly created folder. Run the command **Sudo sh install\_spinnaker.sh** to start the install of the drivers. Follow the installation process (Hit yes on any user prompts that arise) and reboot the computer once completed. Once done, you should be able to type ‘spinview’ into the terminal to open the software required to film and capture images using the camera.

If you have any issues, open up the README from the extracted files and try the following:

* Double check all required dependencies are up-to-date and installed

Spinview has no proper terminal commands, needs c++ to run commands to.

May need to upgrade my ubuntu to 20.04 so I can do dev stuff

Ffmpeg may be required as an alternative to get things running in matlab, if so, move off of guvcview.

May also run a toggle-able option to use spinview instead of guvcview

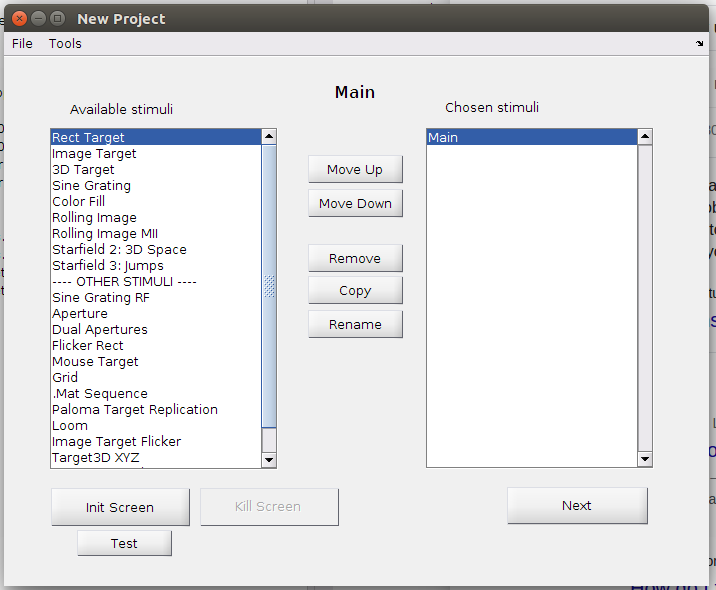
## PS3 Webcam (Legacy)

# Operating Flyfly

## MAIN WINDOW

After starting FlyFly this is what you will see.

Under File you can open a saved file, or save your current file. **NOTE** that you have to save from this main window.

****

## SETTINGS WINDOW

Found under “Tools” in the Main window

*Screen*

We usually run our stimulus computer with at least 2 screens. One is a high resolution display screen (typically Asus or similar), and one is a low-resolution control screen. At the bottom of the section called “Screen” FlyFly tells you what screens are available (in this case 0 and 1). Enter one of these numbers in the box called “Number” and press “Test Init”. This will initialise the screen specified in “Number” for a few seconds, and allow you to make sure that the stimulus is displayed where you want it to be. You can choose Full screen mode or partial screen mode, and if using Partial you can specify the size and width (in pixels) of the screen.

If you are running a three screen setup and you are using three 2650x1440 screens rotated 90 degrees the height of the ‘screen’ should be 4320 and width should be 2560\*. Set the fly position exactly in the centre of the screen, use the set fly position button to get an approximate position and then fine tune the position. Make sure the distance from the monitor is correct and that the Monitor height is set based on the width of the screen and not the height, i.e., if you are using three ASUS pg279qe rotated 90 degrees the height of the monitor will be 60cm.

\*Originally we used 4319 by 2559 due to a prior issue of the screens not initialising but as of 15/03/2022 it was noticed this causes screen tearing with high frequency stimulus as ubuntu doesn’t give full control to flyfly on a partial screen. Instead use 4320 by 2560 or full screen settings in flyfly.

DLP mode should be used if you are using a DLP projector with the colour wheel removed. In this case the R-G-B screens are offset slightly from each other, so PsychToolbox thinks that the stimulus is running at e.g. 120 Hz, but you actually generate grayscale 360 Hz motion from your projector. This feature has not been actively used since 2009, so could be buggy.

*Trigger*

This section is used to separate a small part of the screen from the rest of the stimulus. Instead of displaying the stimulus, the trigger switches between black (OFF) and white (ON). This allows you to attach a photodiode to the screen, and record exactly when the stimulus was on, for high precision offline analysis. The input boxes specify where the trigger is placed, and its size (in pixels). It usually needs some tweaking, as you want it as small as possible so as not to interfere with your experiments, but large enough to give a reliable signal.

Note that the trigger is ON during pre-stimulus time, time, and post-stimulus time (see more on stimuli below), but OFF during pause.

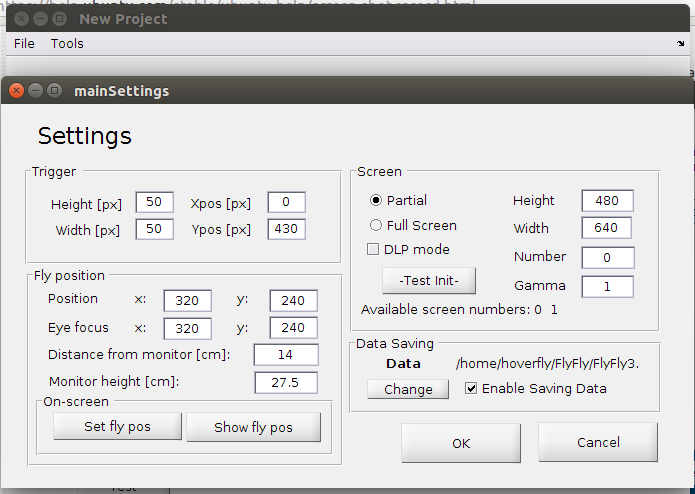
Note that when the trigger is ON, on each frame it fluctuates between white and slightly grey. This can be very useful for troubleshooting precise timing, dropped frame issues, or similar.

Fly position

This specifies where the fly is located relative to the screen. Unfortunately, at the moment it is not used properly, so all perspective distorted stimuli assume that the fly is located centred in the screen (i.e. “Position” and “Eye focus” is in the screen’s centre), so these values are currently not used. However, it is important that you give the “Distance from monitor” and “Monitor height” as this is used by the perspective distorted stimuli.

*Data saving*

Specify the folder where you want the parameter files to be saved to by clicking on “Change”. Ensure that “Enable Saving Data” is ticked.

****

## STIMULUS WINDOW

Once you have selected one or more stimuli from the main window and clicked on Next (or double-clicked on the name of one of the stimuli), the Stimulus Window will appear. The appearance of the window is as below, where we have created five trials of the Image Target stimulus as an example. Each of the sections identified with numbers in red will be discussed below (roughly in order of importance). Note that this is for version 3.1. See further down for important change in version 3.2.

****

### 1. Stimulus Parameters

This area is where the user sets the values of all the *parameters* that determine what exactly the presented stimuli will look like. The parameters are organised in a grid, where each column represents a *trial*, and each row represents a different aspect of the stimulus (a parameter).

Trials are executed sequentially from left to right. The duration of each trial, and whether or not there are pauses between or around trials, are set by the common timing parameters (see below).

In each trial, more than one kind of visual stimulus can be displayed on the screen simultaneously by the use of additional *layers* which run in parallel to each other. Each layer will share the same division into sequential trials.

For information on managing trials, see “Trial Controls” (below). For information on managing layers, see “Layer Controls” and “Layer Manager” (below).

***Stimulus-Specific Parameters***

Each stimulus type has a number of parameters that control how it appears on the screen. For example, in the ImageTarget stimulus shown here, we can control the size of the image displayed on the screen by manipulating the Width or Height parameters. Stimulus-specific parameters are discussed in depth under each of the later sections in this document dedicated to the individual stimulus types.

***Common Timing Parameters***

In addition to the stimulus-specific parameters, there are four parameters in the last four rows of the grid that are common to every stimulus type. These parameters govern the timing of trials. They are given in frames rather than seconds, to avoid rounding off errors.

* **Time -** The number of **frames** the stimulus is displayed on the screen for.
* **PauseTime -** The number of frames between trials. The photodiode trigger will be set to **off** during the pause time.
* **PreStimTime -** The number of frames before the stimulus is displayed on the screen. The photodiode trigger is **on** during PreStimTime.
* **PostStimTime -** The number of frames after the stimulus is displayed on the screen. The photodiode trigger is **on** during PostStimTime.

When multiple layers are used, it is possible to specify different timing parameter values for different layers within the same trial. In this case, FlyFly’s interpretation of these parameters is more complex. See “Layer Manager” for more details.

***Other comments on Parameters***

It is possible to speed up the creation of a stimulus, by preparing the data that goes into the grid using a tool such as a spreadsheet, and copying and pasting data values into FlyFly. The ease with which this can be done depends on the operating system and the spreadsheet tools involved.

In general, the procedure is to:

* Copy a block of cells from spreadsheet software
* Place the cursor inside the stimulus parameters grid at the top-left corner where the copy operation needs to be performed
* Press Control-V to copy (note: even on a Mac, the key required is Control-V, *not* Command-V).

This process works on Ubuntu when copying data values from a LibreOffice spreadsheet. TODO: windows?

On a Mac, this process works for single *rows* of data, but *not* blocks spanning several rows, when using MS Excel.

If you really want to copy blocks of several rows at a time on a Mac, this is possible, but requires:

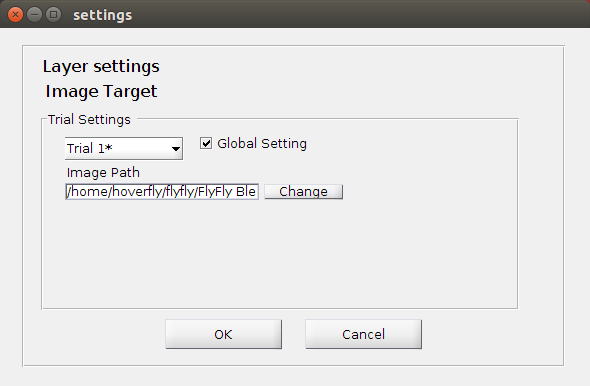
* saving the data in comma-separated (CSV) format
* opening the file with an appropriate text editor
* copying the block
* pasting into FlyFly (still using Control-V, *not* Command-V as might be expected).

However, whether this works or not depends on how the specific text editor handles newline characters. We have not tested a wide range of text editors; TextEdit, XCode and Atom do *not* work, but MacVim and SublimeText do. We suggest trying this out on your text editor of choice.

The opposite process, of selecting a block of cells from the parameters grid in FlyFly and copying it to the clipboard using Control-C (or Command-C on Mac), works on all operating systems, and may be convenient when it is required to, for example, save parameter values back into a spreadsheet.

### 2. Layer Settings

When you click on the “Layer Settings” button, a new window appears, showing the Layer Settings for the stimulus type you have selected (see example below for Image Target).

****

This part of the interface is arguably misnamed: a better name may have been “Stimulus Settings”. These settings are conceptually similar to Stimulus Parameters, and functionally they work more or less the same. The main difference is that stimulus parameters are all numerical values: therefore they can be displayed compactly in a grid, and can undergo operations such as interpolation (see “Interpolate” below). Layer settings deal with non-numeric parameters that are of three different types:

* Checkboxes, corresponding to yes/no decisions
* File paths to files located on the computer
* Strings of text

The example above shows that the Image Target stimulus type has only one setting, namely a path to the file containing the image that will be displayed on the screen. Other stimulus types will have different settings, and these will be described in detail in the sections for the stimulus types. At present, however, most stimulus types have no Layer Settings at all.

In addition to the settings themselves, the Layer Settings window displays two other controls. There is a drop-down box that is used to select the trial for which the layer settings values apply (for stimulus parameters, this would have corresponded to simply choosing the appropriate column in the grid).

In addition, there is a “Global Setting” checkbox. When this is checked, all specified settings values are used for all trials. If you have specified different settings values for different trials, but checked “Global Setting” on, the value specified for the first trial will be used for all trials.

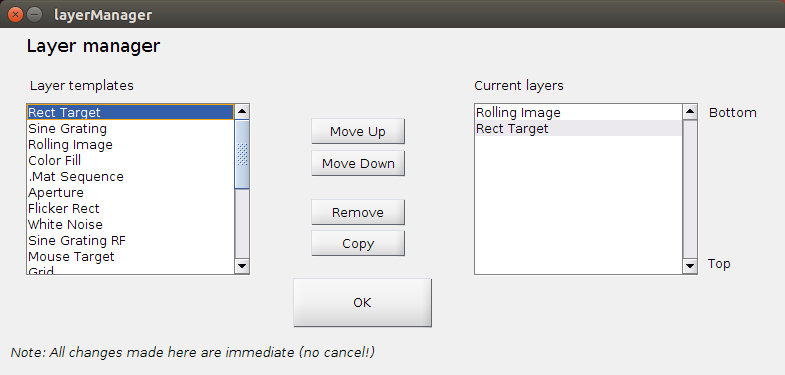
Note – the meaning of “the first trial” is not necessarily the trial specified as “Trial 1” in the Stimulus window, but rather the first trial that is actually run - so if you have specified a trial subset and have set “Global Setting” to on, the layer settings will be taken from the first trial in order in that trial subset (see “Trial Controls” below).

There is no way to mark only some settings as being global and others as trial-specific. There is also no easy way to use a global value, but override it only on some trials (as could be done for stimulus parameters, with the “Use value” function, see “Interpolate” below). Layer settings are therefore less flexible than stimulus parameters.

### 3. Layer Controls

There are two Layer Controls, one on either side of the “Layer Settings” button. The “Layer Manager” button opens the Layer Manager window (see “Layer Manager” below), which allows the user to add multiple layers to the stimulus. Once layers have been added, the drop-down box on the left allows the user to switch from one layer to the next, in order to enter the stimulus parameters and layer settings for each layer.

***Layer manager - finish making changes***

Layer manager allows for the user to display multiple stimuli on the screen at the same time. Double click on the stimulus you want to use from the layer templates (displayed on the left) to move it to current layers (displayed on the right). In current layers, stimulus can be moved up or down the list and copied or removed as required.

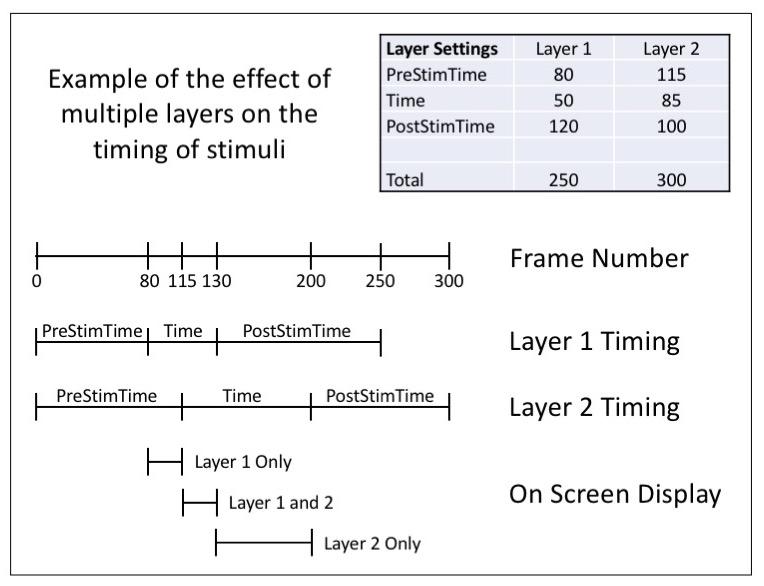
The stimuli are displayed on the screen from bottom to top, ie the stimulus at the top of the list is displayed as the bottom stimulus and the stimulus at the end of the list is displayed on top of all other layers. For example, if you intended to display a small target on a background image, Rolling Image would need to be at the top of the current layers list and Rect Target at the bottom.

***Effect of multiple layers on the timing of stimuli***

When multiple layers are used, it is possible to specify different timing parameter values for different layers within the same trial. In this case, FlyFly’s interpretation of these common timing parameters can be affected by the values entered for other trials.

* **Time -** Each layer is displayed for the number of frames entered for that layer.
* **PauseTime -** The longest pause time listed in all the layers is used.
* **PreStimTime -** The number of frames entered in each layer is the total number of frames displayed before this layer is shown.
* **PostStimTime –** Is affected by the timing of all other layers. The layer with the longest combined PreStimTime, Time and PostStimTime overrides the other layers.

Manipulation of the timing parameters for each layer can be a useful tool in allowing for stimuli in different layers to be shown before or after another layer. For instance, having different PreStimTime settings for each layer allows for the stimulus in each layer to be shown over a different time course. (need to fix wording here) For example, if you intended to display a small target on a background image, where the background motion began before the target appeared on the screen, the layer containing Rect Target would need to have a longer PreStimTime than the layer containing Rolling Image. Explain figure.



The layer manager setting allows two or more stimuli to be displayed simultaneously.

The figure above shows two stimuli with different Timing parameters described in its *Layer Settings* table. The Frame Number image represents the combined parameter values of all layers specified, which in this case is Layers 1 and 2 shown beneath it. The On Screen Display shows only the layer’s Time Parameter periods, highlighting whether the layer is displayed in isolation or in combination with other layers on the visual stimulus screen.

### 4. Trial Controls / Sequencer

The controls on the subpanel entitled “Sequencer” allow the user to add or delete trials. If you enter a number greater than the current number of trials, additional trials are created at the end. Each of the new trials will have its stimulus parameters and layer settings set to the values for the last trial that you currently have. If you enter a number smaller than the current number of trials, then trials will be discarded from the end.

The “Trial Subset” control allows you to run only some of the trials, or to run them in a different order. Tick the checkbox to activate this function, and use the text box to specify the trials that you wish to run. By default, if you enter nothing, all trials will be executed. But it is possible to run trials in other configurations, as shown in the table below.

| Trial Subset Text | Effect |
| --- | --- |
| 1:end | Run all trials (default) |
| 3 | Run trial 3 only |
| 2:5 | Run trials 2 to 5 inclusive |
| [4, 2] | Run trial 4, followed by trial 2 |
| 2:end-1 | Run all trials *except* the first and last trials |
| 2:3:end | Run trial 2 and every third one thereafter (2,5,8,...) |
| randperm(end) | Run trials in a different random order every time |

**NOTE ON RANDPERM!**

**Make sure that you use datamerger modified on July 30, 2020 (function *saveStimulus.m*) , if you use randperm, or the data merging will not take the correct order of trials into account, making your analyzed data look random.**

In general, any valid MATLAB expression can be used here. The string “end” will be automatically replaced by the number of trials at run-time, so it is even possible to use expressions that are not allowable in MATLAB, e.g. the “randperm(end)” example in the table.

Even if the stimulus definition itself has been saved with definitions of all trials, the specific subset used will be correctly saved into the stimulus parameters file at run-time. For example, if you specify a trial subset of [4, 2] as above, then the stimulus definition file will show all trials, but the parameters file saved at run-time will show only two columns in the Params structure, corresponding to trial 4 and 2 respectively.

### **5. Execution Controls**

This block of controls is used for running the stimulus. The “Run” button is especially convenient for quickly seeing what a stimulus looks like. “Run” will initialize a screen according to the specification in the General Settings, run the stimulus and automatically close the screen at the end. Afterwards, the stimulus info will be saved into the directory as chosen by the user in the ‘settings’.

However, when actually conducting an experiment, it is preferable to follow a three-step process:

1. Click on “Init Screen” to initialize the screen at the start of the experiment
2. Click on “Run” to deliver the stimulus
3. Repeat until you have done all your desired experiments
4. Click on “Kill Screen” to close the screen at the end of your experimental session

The reason for this is to allow the experimenter to perform all other tasks needed before running the stimulus, e.g. entering comment markers into data acquisition software, and also to ensure that the screen will in fact display correctly. The photodiode trigger remains on-screen in the “off” state and will be recorded; this also facilitates identifying the data block in the recorded trace.

**Note for using the rotated screen version of Flyfly:**

When using a connected camera during the rotated screen version, if the camera is successfully connected, the “Init Screen” button will initialise the camera software required, and the “Run” button will set the camera to record for as long as the stimulus is running.

**End note.**

The “Grid” button allows the user to see a grid overlaid on the screen, showing pixel coordinates of various locations on the screen. This may be useful in designing a stimulus, estimating the rough screen location of a receptive field during recording, etc. “Grid” only works after the user has clicked on “Init Screen”. Clicking on the “Clear” button removes the grid again. Note that the grid does NOT remain on-screen once the stimulus starts running; in order to obtain this effect, it is necessary to add a layer containing the “Grid” stimulus (see below).

### 6. Stimulus Navigation Controls

If you have defined more than one stimulus for an experimental session, you can navigate from one stimulus to the other via the “Previous” and “Next” buttons, or by directly selecting the stimulus name from the dropdown box. (Note that this is “stimulus” in the sense of a series of layers displayed over a number of trials, rather than the individual stimulus displayed in a particular layer in one trial.)

### 7. “Interpolate” / Editing Controls

These controls allow the user to more conveniently enter values into the stimulus parameters grid.

The “Row/Col” boxes provide information only - they display the indexes (row and column) of the cell in the grid where the cursor is currently located. (It is possible to edit these values, but this has no effect.)

The “lin” and “log” buttons allow interpolation of values. By default, interpolation operates on the current row only, and sets the values in the row to an interpolation between the first and last values in the row. The interpolation is linear if “lin” was selected (adding a constant each time), and logarithmic if “log” was selected (multiplying by a constant each time). If the “All rows” checkbox is ticked on, the interpolation is performed on all rows instead of just the current row.

The “Use value” button takes the value that is currently under the cursor, and copies it into other cells of the current row. The behaviour of “Use value” is controlled by the text box to its right, in which the user specifies the period of repetition of the value. If the user specifies a value of 2, the value is copied into every second cell both before and after the current cell (so that if the current cell was in column 9, the value in that cell would be copied to columns 1, 3, 5, 7, and 11, 13, …). With the default value of 1, the value under the cursor is copied to all columns in the current row. “Use value” is modified by “All rows” - if the checkbox is selected, the whole current column is copied.

The “Shuffle” button randomly shuffles the values in the current row into a different order. This button is also modified by “All rows”, so that selecting the checkbox will shuffle values in all rows. It’s important to note that the rows will be shuffled *independently*, i.e. clicking “Shuffle” in this case does *not* have the effect of shuffling entire columns (trials) around. To shuffle whole trials at a time, use something like the “randperm(end)” technique in the Sequencer.

The two remaining buttons “<<” and “>>” shift the entire current column to the left or to the right.

### 8. Screen Info

When a screen has been initialized, information about the screen is displayed here. The information shown is:

* Screen: The resolution in pixels, and the screen frame rate
* Skipped frames: the number of frames skipped in the last stimulus that was run
* Skipped total: the running total number of frames skipped so far
  + Clicking on the “Reset” button sets the Skipped total back to zero

### 9. Impulse checkbox

Ticking this checkbox allows a stimulus to be presented as an impulse stimulus. This means that only the first frame of each trial is carried out. After the first frame, all visual components of the stimulus remain stationary on the screen until the trial completes.

The impulse checkbox applies at the layer level only, so that it is possible to display one layer that moves continuously while the other is an impulse stimulus.

If the first frame entails performing a motion, the use of Impulse allows the user to define a “jumping” motion across trials. This is true with, for instance, the Starfield 2 stimulus, in conjunction with setting “Retain into next trial” to 1 (see description of the stimulus type below).

### 10. Stimulus and Layer Name Controls

These controls allow the user to change the name of the current stimulus (block of layers x trials), as well as the current layer, by clicking on the “Edit” buttons next to the names. This is useful for the sake of documenting the structure of an experiment, for example when there are multiple stimuli with the same default name (taken from the stimulus type you selected when creating the stimulus), or multiple layers with the same default name. Stimulus and layer names are displayed in the stimulus selection dropbox (see “Stimulus Navigation Controls” above”) and the layer selection dropbox (see “Layer Controls” above). Stimulus names are also displayed when you return to the Main window.

### 11. “File” menu

This menu offers the same options as the File menu from the Main window. However, there are known bugs associated with using the File menu options inside the Stimulus Window, and this menu SHOULD NOT BE USED until these bugs have been fixed. Instead, create and save experiment definition files using the corresponding options from inside the Main window only.

### 12. “Tools” menu

There are two menu options on this menu. “Settings” is identical to the Settings option in the Main Window. The specific menu item for a stimulus is the “Record Stimulus” option. This allows the user to save all frames rendered in a stimulus, in one of two ways: as an .avi movie, or as a sequence of images. To use it, do the following:

* Initialise the screen
* Select Tools > Record Stimulus > [either “AVI movie” or “Image sequence”]
* Specify the frame rate at which frames will be sampled (use the screen frame rate if you want to capture every frame)
* In the case of “Image Sequence”, you will be prompted for a name to give to the folder where the images will be saved
* Finally, you will be prompted to select the location where the .avi movie or folder will be saved
* Note that the stimulus will not actually be displayed on the screen as well. It will only be saved in the format you have selected.

### 13. “Special” menu

DO NOT USE the options in this menu. They are no longer available, and the menu will be removed in a later release.

### 14. M-sequence controls

M-sequences (or maximum-length sequences) offer a way to generate a pseudo-random sequence of values. In FlyFly, the user can specify that the values of the current row should be taken from an m-sequence containing either 2 or 3 equally spaced values. The two edit boxes on the Stimulus Window define the range of values, in conjunction with the “Base value” box in the subsequent m-sequence dialog window. The left box defines the central value of the m-sequence (we can call it *C*), and the second box defines the spacing between values (call it *s*). If “Base value” is equal to 2, a binary m-sequence is generated, and all values will be either *C+s* or *C-s*. If “Base value” is equal to 3, the sequence is ternary, and *C* will also be one of the possible values.

Examples:

* Set the left and right box to 0.5 and 0.5, and Base value to 2, to generate an m-sequence of 0s and 1s.
* Set left box to 0, right box to 1, Base value to 3, to generate a sequence made up of -1, 0 and 1.

It is possible to specify additional details of the exact m-sequence to use, by modifying the other 3 values in the m-sequence dialog box, but detailing how this works falls outside the scope of this manual. The user should consult appropriate references on how m-sequences work before using them. TODO: eventually we will write something more here.

### 15. Gaussian White Noise controls

These controls are used to add Gaussian white noise to the values in the currently selected row. FlyFly will generate samples from a Gaussian distribution, with mean of zero and sigma equal to the value you specify in the “Variance” box, and add the sample values to the values that are in the current row. Note that the label “Variance” is actually incorrect; this value represents the sigma or standard deviation. By setting all values in the current row to the value *m*, it is therefore possible to generate samples from the distribution with sigma as specified and mean equal to *m*. Alternatively, use different starting values in the current row to sample from distributions with different means.

Where are they saved?

### 16. Paloma XY generation

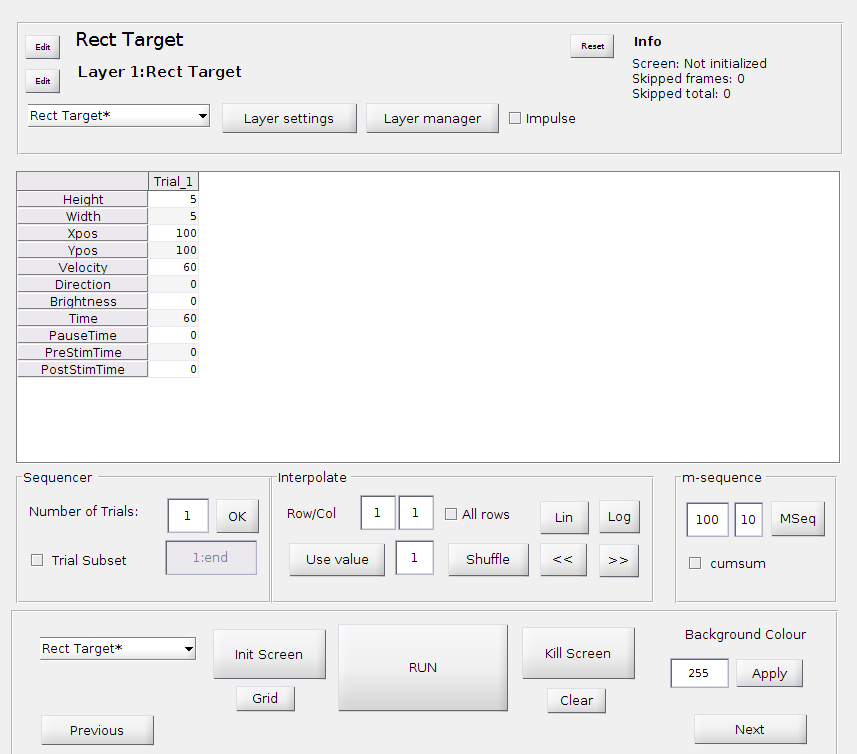
These controls are used only by the “Paloma Target Replication” stimulus and are described in the section for that stimulus. The controls are expected to be removed from FlyFly in a later release.

### 17. High Definition Image

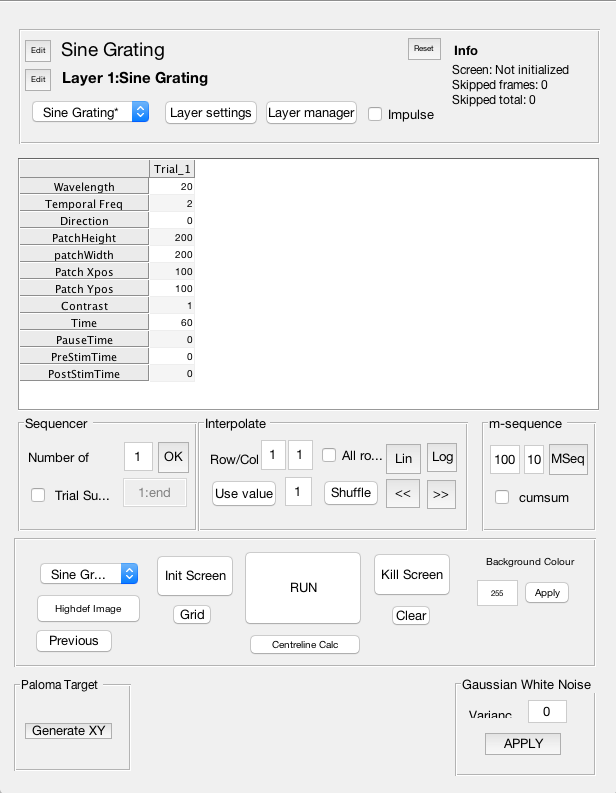
*NOTE: For Flyfly camera operations, the camera must be connected* ***prior*** *to boot. If the camera is connected after boot, make sure to reboot the system prior to using Flyfly.*

This button allows for the capture of a high detail image that is needed for calculating an accurate centreline position of the tethered animal. It will open image capturing software Guvcview, take the image, and close after 10 seconds.

### Version 3.2 change

Note that as of version 3.2 the user can set the background colour (the screen colour when no stimulus is shown) from the stimulus window, on a scale from 0 - 255. The background colour is applied by either pressing “Apply” or “Run”, and is saved with the other parameters.

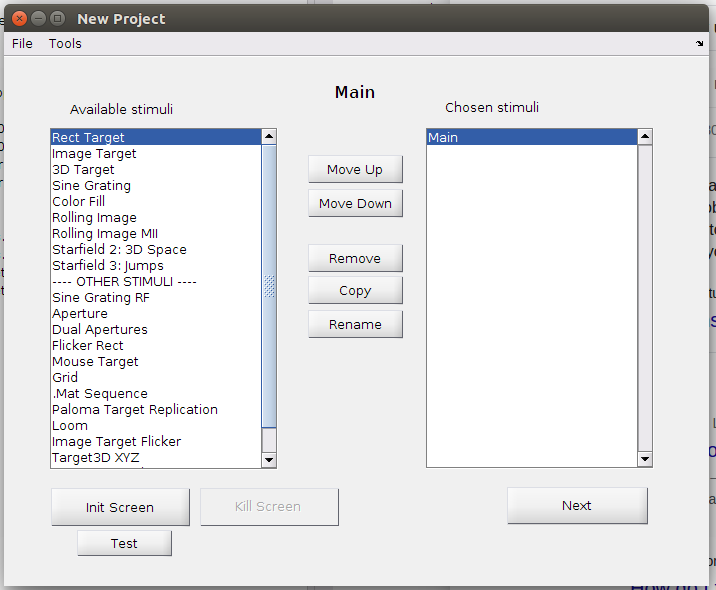
### Version 3.3 Change



**Note:** In version 3.3 init screen launches guvcview for capturing footage that is used with Deeplabcut. Currently pauses dont work properly with recording.

# MAIN WINDOW: STIMULI

In the main window you will see a list of the *Available stimuli.* You choose what stimuli to use by clicking on their name in the left hand column. They are then added into the right-hand column (*Chosen stimuli*). You can change the name of any stimulus by using the *Rename* button. You can change the order of your chosen stimuli by using the *Move Up* and *Move Down* buttons. In addition, if you have a very complicated stimulus, you can use the *Copy* button, which copies all parameter settings. This, together with *Rename*, can be very useful.

****

## RECT TARGET

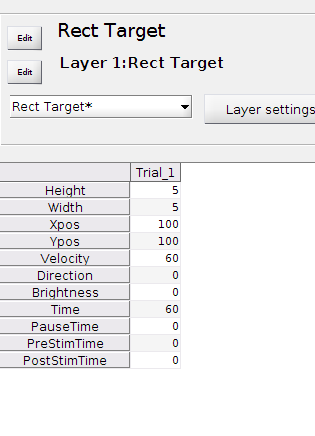
The Rect Target stimulus displays a rectangular target. Its size is given by *Height* and *Width* (in pixels). Its starting position is given by *Xpos* and *Ypos.* This is given in pixels and refers to the center of the target.

If you want the target to move, you specify its *Velocity* in pixels/s. This can be in either positive or negative values. Negative velocities move in the opposite direction to positive velocities.

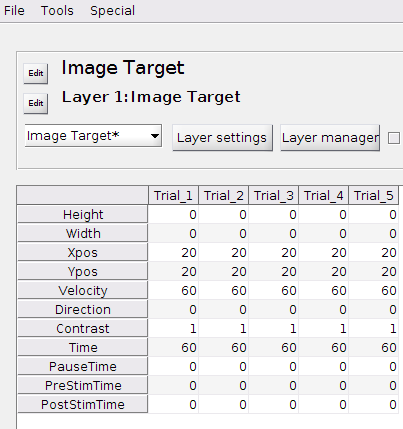
For moving targets you specify its *Direction* in degrees, where 0 is right, 90 up, 180 left and 270 down.

You can change the *Brightness* of the target on a scale from 0 (black) to 255 (white).

*Time*, etc, have been described above.

****

## IMAGE TARGET (NB - Sarah needs to check/fix highlighted parts)



This stimulus displays a static or moving image on the screen. An example of its use is size tuning for TSDNs using circular targets, from 1 pixel to full screen height.

The *Height* and *Width* sets the size, in pixels, of the image being displayed, however the aspect ratio of the original image is always maintained. If both the height and width are set to 0 the image is the default size (ie the size of the original image). If the height is >0 and the width is 0, the image is set to the height entered while the width is determined by the original aspect ratio of the image. Conversely, if the width is >0 and the height is 0, the image is set to the width entered while the height is determined by the original aspect ratio of the image.

When both the height and width are >0, the image is set to whichever setting is lower and the other setting is determined by the original aspect ratio of the image. Negative values will cause a crash.

The *Xpos* and *Ypos*gives the starting location of the middle of the image being displayed. If you want an image to start off the screen you need to compensate for its size in determining the start position. Both negative values and values outside the screen dimensions work perfectly well. Note - Xpos = 0 denotes the left-hand side of the screen and Ypos = 0 denotes the top of the screen.

*Velocity*sets the speed, in pixels per second, at which the image moves across the screen. Both positive and negative velocities work well, with negative velocities moving in the opposite direction to positive velocities, set at 0 the image is stationary. Ensure you check that the velocity being used is not too quick for the refresh rate of the monitor the stimulus is displayed on. Small images being displayed at fast speeds may not be presented as a smooth movement across the screen but instead may appear as if the image is jumping positions between frames.

The *Direction* sets the heading in degrees, in which the image moves across the screen. Note - 0 = right, 90 = down, 270 = up and 180 = left.

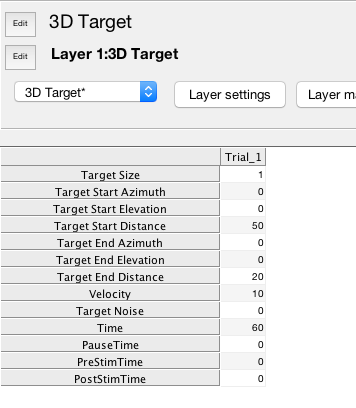
The *Contrast* is scaled between -1 and 1, with number close to 1 being darker and numbers close to -1 being lighter.

You need to use the *Layer settings* to specify the path to the image that you want to display. This stimulus often defaults to the default image (especially if trying to only

show a subset of trials), so double and triple check that the image you chose is actually shown on the screen, in all trials.

Note – acceptable image formats are png….???

## 3D TARGET



This stimulus is a 3D counterpart to “Rect Target”. It displays a spherical object moving

through 3-dimensional space in a straight line between two specified points.

An important feature that needs to be kept in mind is that 3D Target does not work like the

rest of FlyFly. Other stimuli use the Time field to determine the number of frames

that should be displayed per trial. In the case of 3D Target, we specify the start and end

positions, plus the velocity of movement, and ***this combined information determines the***

***duration of the stimulus***. ***Note that the Time field is still displayed in the 3D Target parameter grid, and can be edited, but supplying a value here will have no effect***.

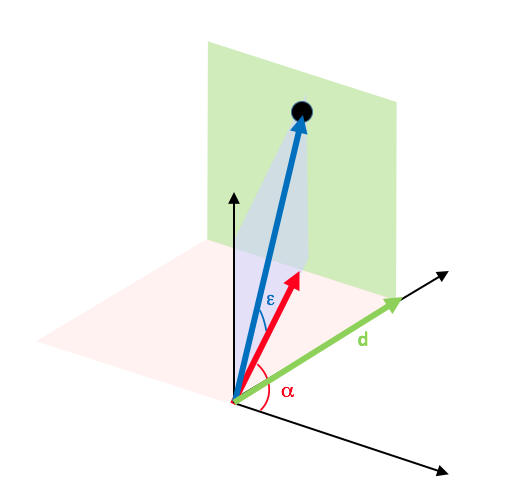
Once a start and end position and a velocity have been specified, the target will move in a straight line from the start, at the desired velocity, until it reaches the end. See next page for an illustration of how to specify start and end position. Velocity is measured in cm/s.

If you specify a value n for Target Noise, the target position in every frame will deviate from its “correct” position, by a random amount between 0 and n cm in each of the x, y and z directions. This can be used to make the target follow a more “jittery” path.

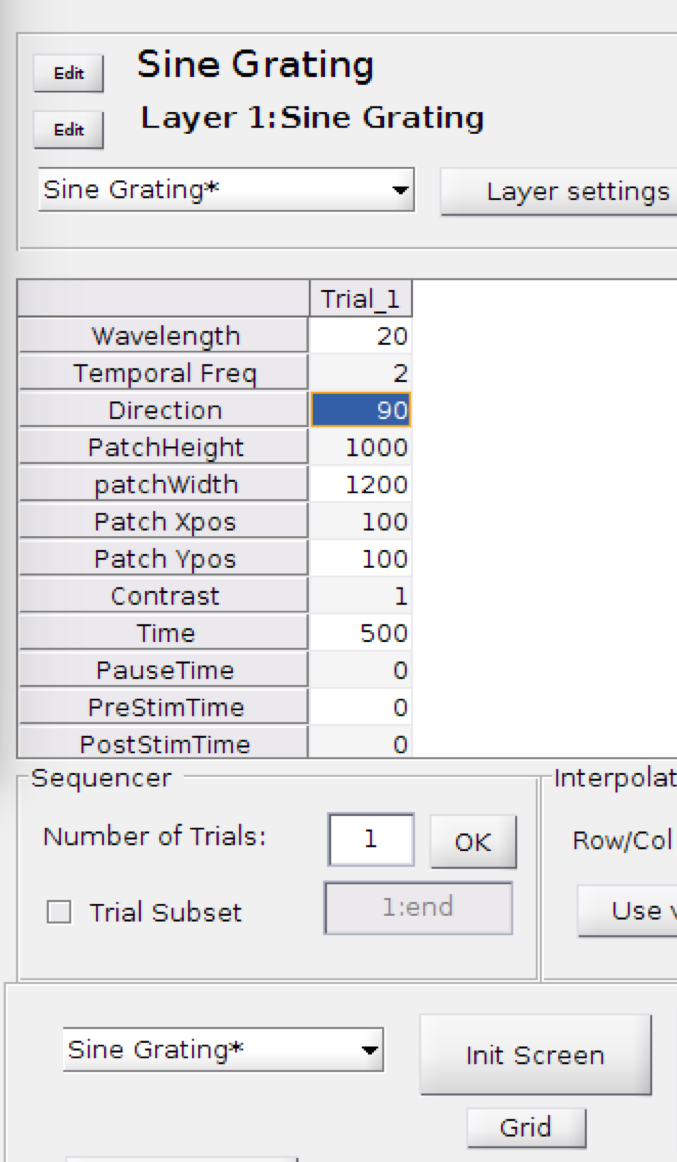
Specification of the start and end positions is done by providing an azimuth and elevation value, as shown in the diagram below.

The target is located at the position of the black sphere. In order to “get there”, we provide three steps that need to be carried out, which are represented by azimuth, elevation and distance.

The azimuth angle is the degree of turning measured in the XZ plane, which you can think of as the “yaw angle” by which you need to turn your head to look in the right direction. The azimuth angle ⍺ is shown in red. The elevation angle is the “pitch angle” by which you need to lift (or turn down) your head after performing the yaw turn, and is shown as the angle ε in blue. Finally, you need to specify distance, which in Target 3D is measured as the Z “depth” value, i.e. the projection of the target’s position vector onto the Z axis (NOT the straight-line distance from the origin to the target). Distance is shown as d (in green). Distances are measured in centimetres.



## SINE GRATING



This stimulus displays a moving sinusoidal grating. The stimulus settings allow the contrast of the sine-wave to be varied across *Wavelength* (in pixels), *Direction* of movement (in degrees), *Temporal frequency* (in Hz) and *Contrast* (between 0 and 1).

*Direction*

0 = right

90 = up

180 = left

270 = down

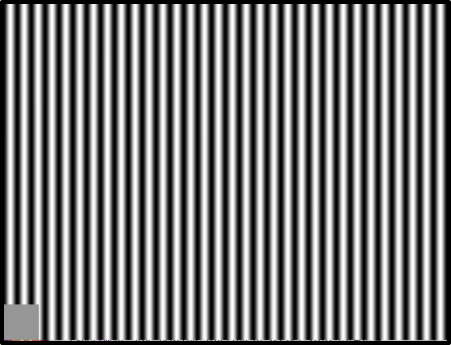
-90 = down

*Contrast*

1 = full contrast, using the full range of the screen from white to black

0 = compressed completely towards the mean value of the screen (in our case 127)

The portion of the screen the sine wave occupies is controlled by manipulating “*PatchHeight*” and “*patchWidth*” (both in pixels). The image below depicts a small sinusoidal grating in the top left corner has a patch height of 200 and patch width of 200 pixels, whereas the one next to it has patch height and width of 1000 and 1200 pixels.



Sine grating positioning can also be varied around the x and y axis of the screen by using the “*Patch Xpos*” and “*Patch Ypos*” settings, both in pixels. The position refers to the center of the patch.

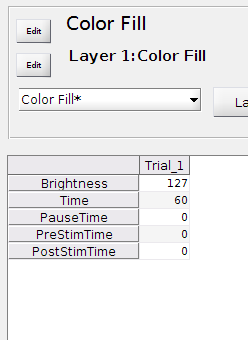
**NOTE** that the maximum temporal frequency should be given by the temporal resolution of the screen. For example, if you run your stimuli at 160 Hz, don’t use a temporal frequency above 40 Hz.

## COLOR FILL

This stimulus has previously been used to set the background color between trials, but has now been replaced with the *Background Color* function.

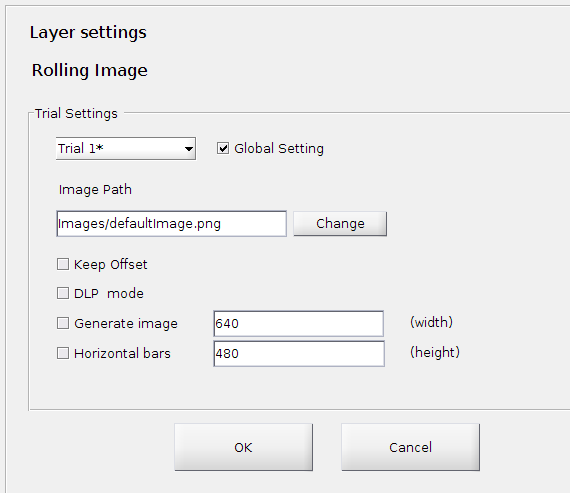
Color fill has a unique bug/feature where it does not switch off like other stim do. Stimulus duration has no effect unless the duration is longer than the duration of all other layers. Once it switches on it will only switch off when the entire sequence of trials is finished.

Also make sure color fill is on the top of your layer manager, otherwise it can occlude your other stimuli.

****

## ROLLING IMAGE

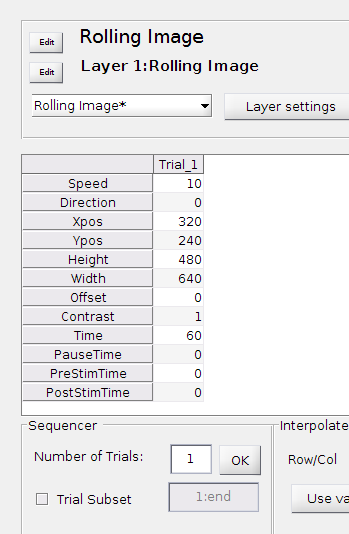
***This stimulus is very buggy, so please double and triple check every new stimulus, and every single parameter before trusting it!!!!!***



This stimulus displays an image sliding across the screen. The image is set using the *Image Path* in *Layer settings*. In layer settings you can also generate an image with random black and white bars by clicking “*Generate imag*e”. If you want the bars to be horizontal, click “*Horizontal Bars*”.

If using a DLP projector, ensure that *DLP mode* is chosen.

*Keep Offset* means that the position of the image is retained between trials.



The main window of Rolling Image gives you control of the *Speed* (pixels/s, negative values give you motion in the opposite direction), *Direction* (degrees) and position of the stimulus (*Xpos* and *Ypos*, both in pixels).

*Direction* can only have values between -90 (down) and 90 (up), other values will make FlyFly crash.

**NOTE that if you use Direction 0, Xpos and Ypos refers to the center of the image. However, if you rotate the motion and use e.g. Direction 90, the location of the stimulus ALSO changes, and rotates around its top left corner. This has caused severe headaches in the past so ensure that you check your stimuli carefully!**

*Offset* refers to what portion of the image is shown at the start. If 0, it starts from the left hand side of the image. If 100, it starts displaying the image 100 pixels in. This is overruled if using *Keep offset* from *Layer settings*.

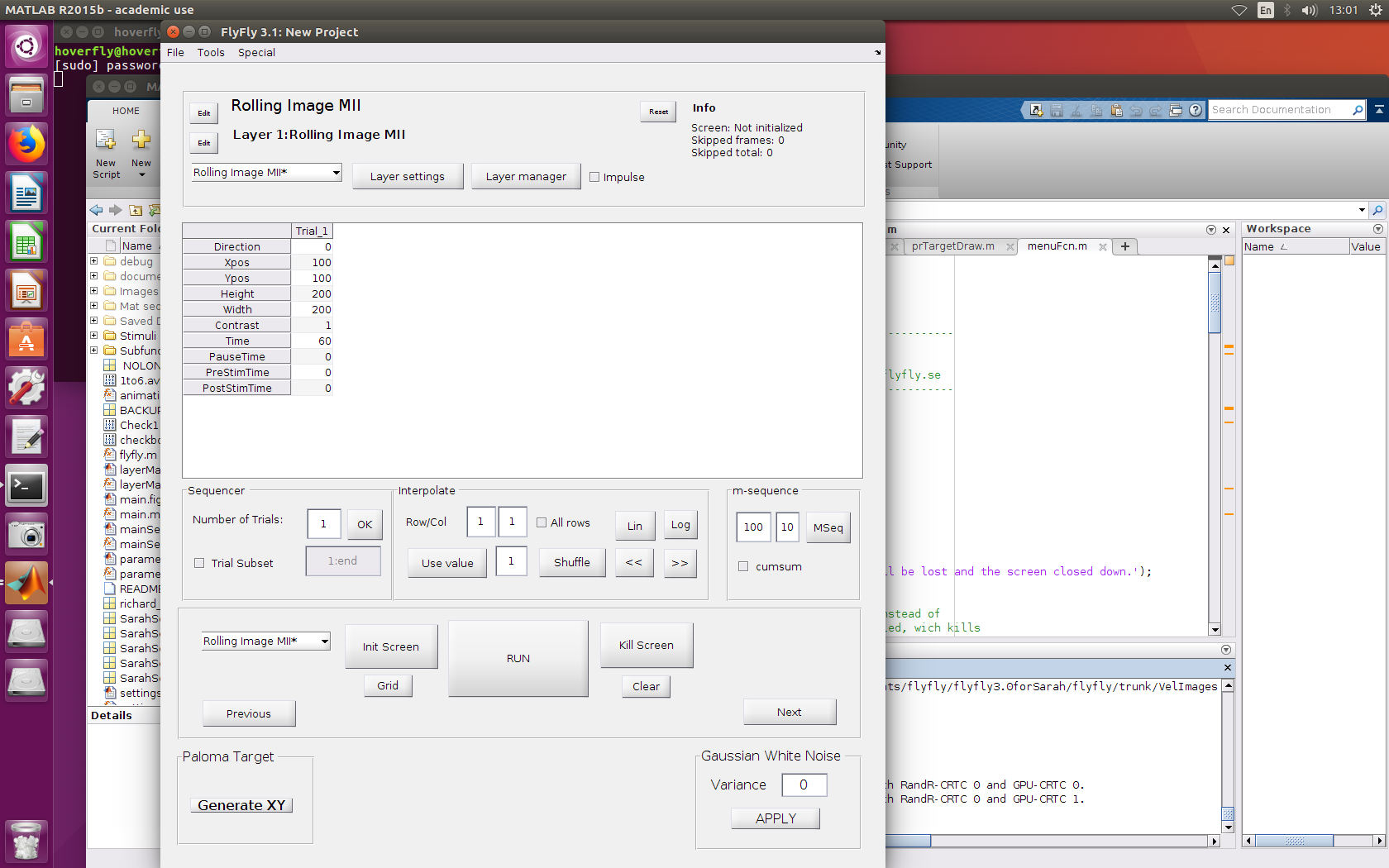
*Contrast*

1 = full contrast, using the full range of the screen

0 = compressed completely towards the mean value of the screen (in our case 127)

## ROLLING IMAGE MII

***This stimulus is very buggy, so please double and triple check every new stimulus, and every single parameter before trusting it!!!!!***



This stimulus displays an image, sliding across the screen. The *Direction* can only have values between -90 (down) and 90 (up), other values will make FlyFly crash. 0 is motion to the right. You have to combine the *Direction* with the motion (given in Layer settings) to get the image to move to the left.

The *Height* and *Width* tell you how much of the image is shown. If the values are larger than the image, it will simply ignore what you entered.

In theory, the *Xpos* and the *Ypos* give the location of the center of the image, but in practice, this is where most problems arise. For example, if you use a *Direction* of -90 or 90, the image will rotate around the top left corner of the image, and actually switch where it is located on the screen. In addition, if you change the *Height* and the *Width* of the image, this also affects the position of the image on the screen, in what appears to be random ways.

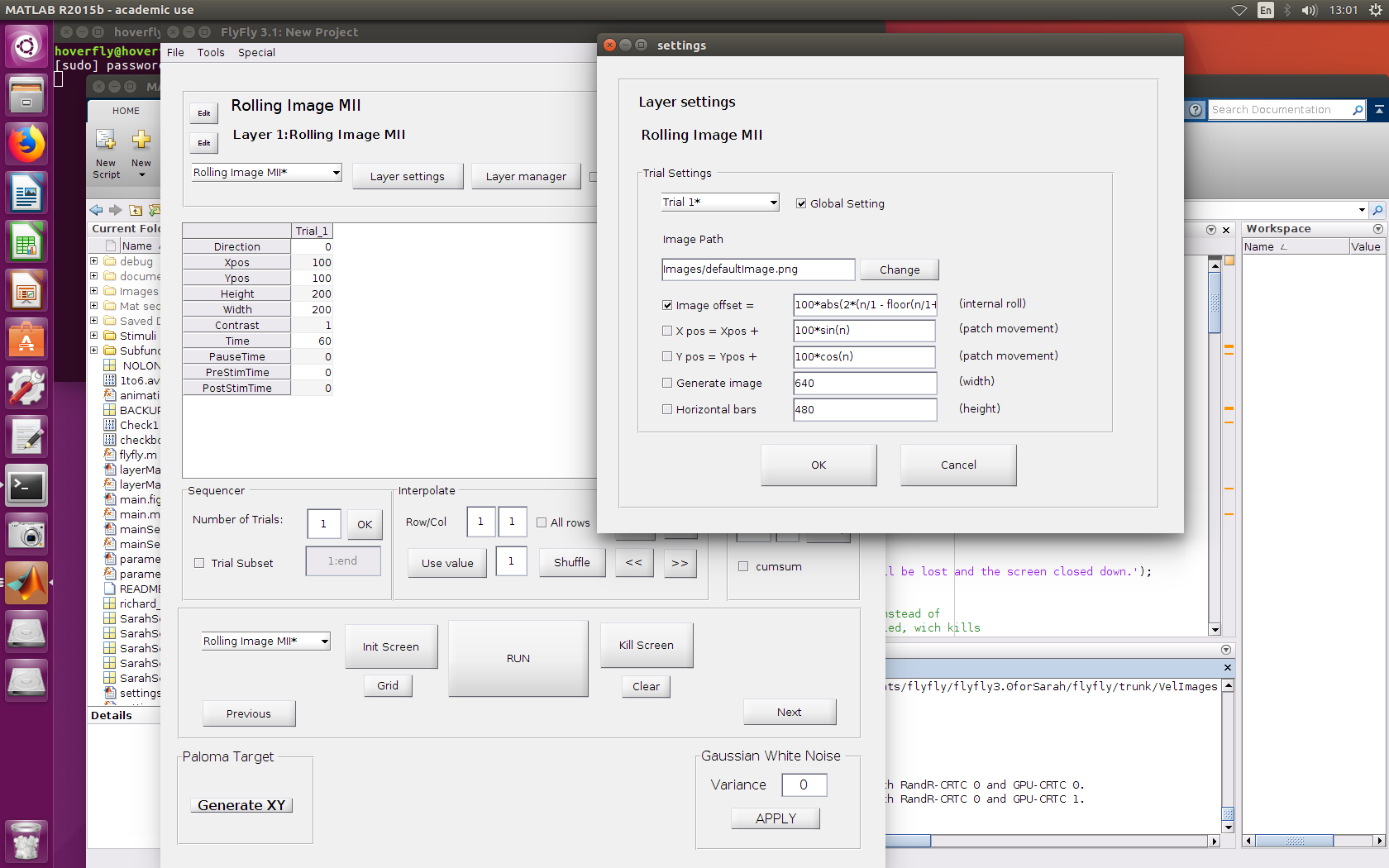
The *Contrast* is scaled from 0 to 1, but you can enter values outside of this range without the software crashing. It simply performs some apparently self invented calculations.

You need to use the *Layer settings* to specify the *path* to the image that you want to display. This stimulus often defaults to the default image (especially if trying to only show a subset of trials), so double and triple check that the image you chose is actually shown on the screen, in all trials.

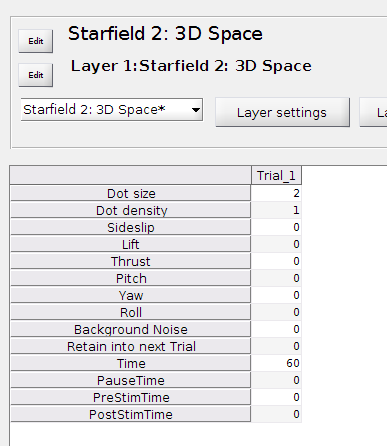
In *Layer settings* you can make the image move in some pretty cool ways, which is what the stimulus was designed for. *Image offset* specifies what part of the image is displayed. For example, if I only show 200x200 pixels of an image that is 2048x1240 large, and then add *Image offset*, the location of the patch itself will be stationary on the screen, but within the patch the pattern will move.

Similarly, *Xpos* and *Ypos* (patch movement) can be used to specify the motion of the patch itself. It can move over a stationary image, or an image that has its own Image offset. For all 3 options (*Image offset, Xpos* and *Ypos) n* specifies each frame. Try different combinations, and you will soon get the idea.

The last 2 options in *Layer settings* give you the opportunity to generate a random image with vertical (*Generate image*) or *Horizontal bars*. The bars have random brightness of 0 (black) or 255 (white). The values given specifies the width, or the height of the bars, respectively, in pixels.



## STARFIELD 2: 3D Space

****

The Starfield represents a 3-dimensional space in which a number of black spheres are suspended. The spheres can move together in unison, simulating self-motion on the part of the fly. The Starfield therefore provides a way to simulate optic flow in 3 dimensions.

The Starfield stimulus maintains an internal representation of the underlying modelled 3-D space and the positions of the spheres within that space. Nominally, the space is modelled as a cuboid with sides 4m in length in each of the x, y and z directions (but see “Note on movements at high velocities” below).

Spheres are displayed on-screen as flat black circles. All spheres are modelled as being the same size (as specified by the “*Dot size*” parameter, given in cm), and are scattered randomly through 3-D space, so that some are closer to the viewer than others.

Distance from the viewer is visually conveyed so that size is inversely proportional to distance away from the viewer.

The Starfield can be made to move in 3-D space, with all spheres moving together. The view on the screen is the projection, onto 2 dimensions, of how the underlying 3-D space would appear to an idealized viewer located at the position of the fly’s head[[1]](#footnote-0).

“*Dot size*” specifies the size of the sphere in the underlying modelled 3-D space, given in cm.

This number is NOT directly interpretable as numbers of pixels on the screen, or centimetres on the screen. Instead, the number of pixels occupied by a sphere will increase as the sphere moves closer to the fly.

“*Dot density*” refers to the density of the spheres in the underlying space. Because the space is finite, increasing the density has the effect of increasing the overall number of spheres in the space, and hence the number of circles displayed on-screen. Density is proportional to the probability that a sphere will be found within a certain 3D volume. If you enter 1, this means a density of 100/m3.

“*Sideslip*”, “*Lift*” and “*Thrust*” allow the user to specify translational movements of the starfield, in respectively the left-right, up-down and forward-backward directions in relation to the screen. These values are specified as *velocity* of movement, in units of cm/s. In particular, therefore, these values do NOT give the *amount* of movement performed during a trial; instead, the amount of movement is determined by the velocity in combination with the duration of the trial. Also, this is the velocity of motion of the underlying *simulated* 3D space, and does not refer to centimetres on the screen.

“*Yaw*”, “*Pitch*” and “*Roll*” are used to specify rotational movements of the starfield, as angular velocities with units of degrees/s. It is important to **note that rotations *cannot* be combined in Starfield, as doing so will not result in the correct movement**. It is fine to combine translations with either rotations or other translations.

“*Background noise*” adds random noise to each frame of the trial. For a value of 1, a small increment is added to or subtracted from the amount of motion to be carried out in that frame, up to 100% of its original magnitude. For a value of 2, the increment can be up to 200% of the original value, etc.

By default, when a stimulus consists of multiple trials, a new Starfield is initialized at the start of each trial. Sometimes, it is desirable to create a longer trajectory over time of the same Starfield, by using a sequence of trials with a different motion in each trial. In such cases, it is necessary to keep the same Starfield across all trials. In order to achieve this, the “*Retain into next Trial*” parameter needs to be set to 1 for every trial that retains the starfield from the previous trial (in the position where it ended up at the end of the trial).

**Note on movements at high velocities:**

Because the 3-D space is finite, and because it can be moved relative to the viewer, there exists a risk of carrying out a movement that causes the stimulus to “run over the edge” of the space. ***Note that in this event, the stimulus will “crash” completely.***

There are mechanisms in the code that attempt to avoid this situation: when Starfield is asked to perform a motion that shifts the current viewpoint beyond the boundaries of the space, the spheres behind the current viewpoint are “moved across” to appear in front of the viewpoint. In this way, space is continually “rolled into place” in front of the viewer.

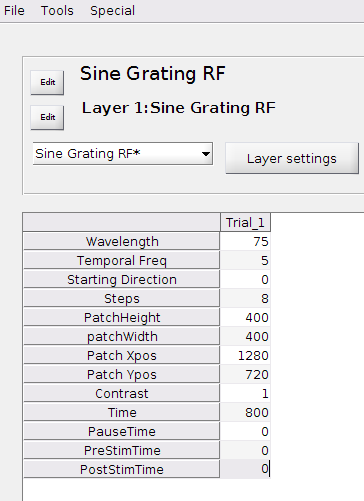
However, there are two side effects of this mechanism: (i) if the Starfield moves along a prolonged trajectory in one direction, the 3-D space will continually “repeat itself”, as the same points are repeatedly rolled into place again and again. Space is effectively made circular, and it is possible that the cyclic nature of the stimulus can have unintended effects on the results; (ii) when the velocity of movement approaches or exceeds the size of the cuboid, i.e. for velocities close to 200 cm/s or more, this mechanism will fail anyway.

If a Starfield stimulus is required to translate at high velocities (in excess of 150 cm/s), it is recommended that the user should perform a “dry run” of the stimulus first before starting a recording. For lower velocities, the possible effect of presenting a repeating trajectory through a cyclic space, as discussed above, should be considered before using the stimulus.

## STARFIELD 3: JUMPS

This stimulus is currently not used

## SINE GRATING RF



This stimulus was designed for mapping the receptive field for responses to sinusoidal gratings.

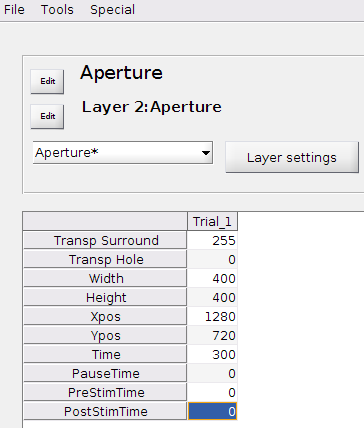
*Wavelength* in pixels

*Temporal Freq* in Hz

The *Starting Direction* sets the first direction, in degrees, in which the sine grating moves, while the number of *Steps* determines how many directions are shown. For example, if *Steps* is set to 8, Starting Direction to 0 and Time to 800, the sine grating in the patch will be shown in 8 directions (0, 315, 270, 225, 180, 135, 90 and 45) for 100 frames each. Note - 0 = right, 90 = down, 270 = up and 180 = left. The default setting presents the directions in a clockwise order, however this can be changed in *Layer settings* to a counter clockwise order.

The size of sine grating patch is given by the *PatchHeight* and the *patchWidth* (in pixels) and the location in *Patch Xpos* and *Patch* *Ypos* (in pixels, referring to the centre of the sine grating patch). Use multiple trials each with a unique position on the screen to cover the entire screen of the portion of the screen to be mapped. The *Contrast* of the sine grating is given on a scale from 0 (the entire patch is grey) to 1 (the darkest point of the bar is black and the lightest point between the bars is white).

## APERTURE



This stimulus is designed to be used as *Layer 2*, allowing you to either show or cover a section of the stimulus in *Layer 1*.

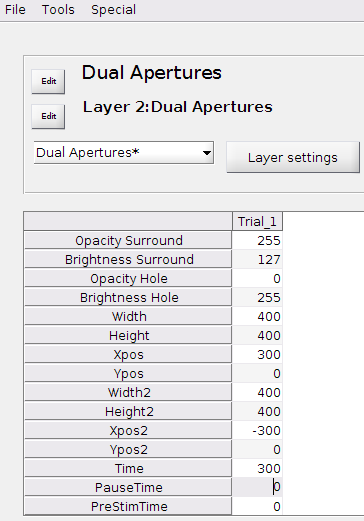
*Transp Surround* is the transparency of the area surrounding the aperture, it is scaled between 0 and 255, with 0 being completely transparent and 255 solid black. Likewise, *Transp Hole* is the transparency of the aperture also scaled from 0 to 255, however in this instance 255 is solid white. In this stimulus you are unable to change the brightness of either the surround or the hole, whereas in Dual Aperture both the brightness and the transparency can be adjusted.

The size of the aperture is given by the *Width* and the *Height* (in pixels) and the location by *Xpos* and *Ypos* (in pixels, referring to the centre of the hole). Note - Xpos = 0 denotes the left-hand side of the screen and Ypos = 0 denotes the top of the screen.

The default setting is for the aperture to be an ellipse, however this can be changed in *Layer settings* to a rectangle.

WARNING: Using Aperture in a multiple screen setup is known to cause heavy frame drops after a non specified amount of time. This is possibly due to Aperture not releasing the memory it uses each sequence, therefore causing memory usage to skyrocket.

## DUAL APERTURES to be completed by Sarah



This stimulus is designed to be used as *Layer 2*, allowing you to either show or cover up to two sections of the stimulus in *Layer 1*.

X Pos Y Pos = centre of image

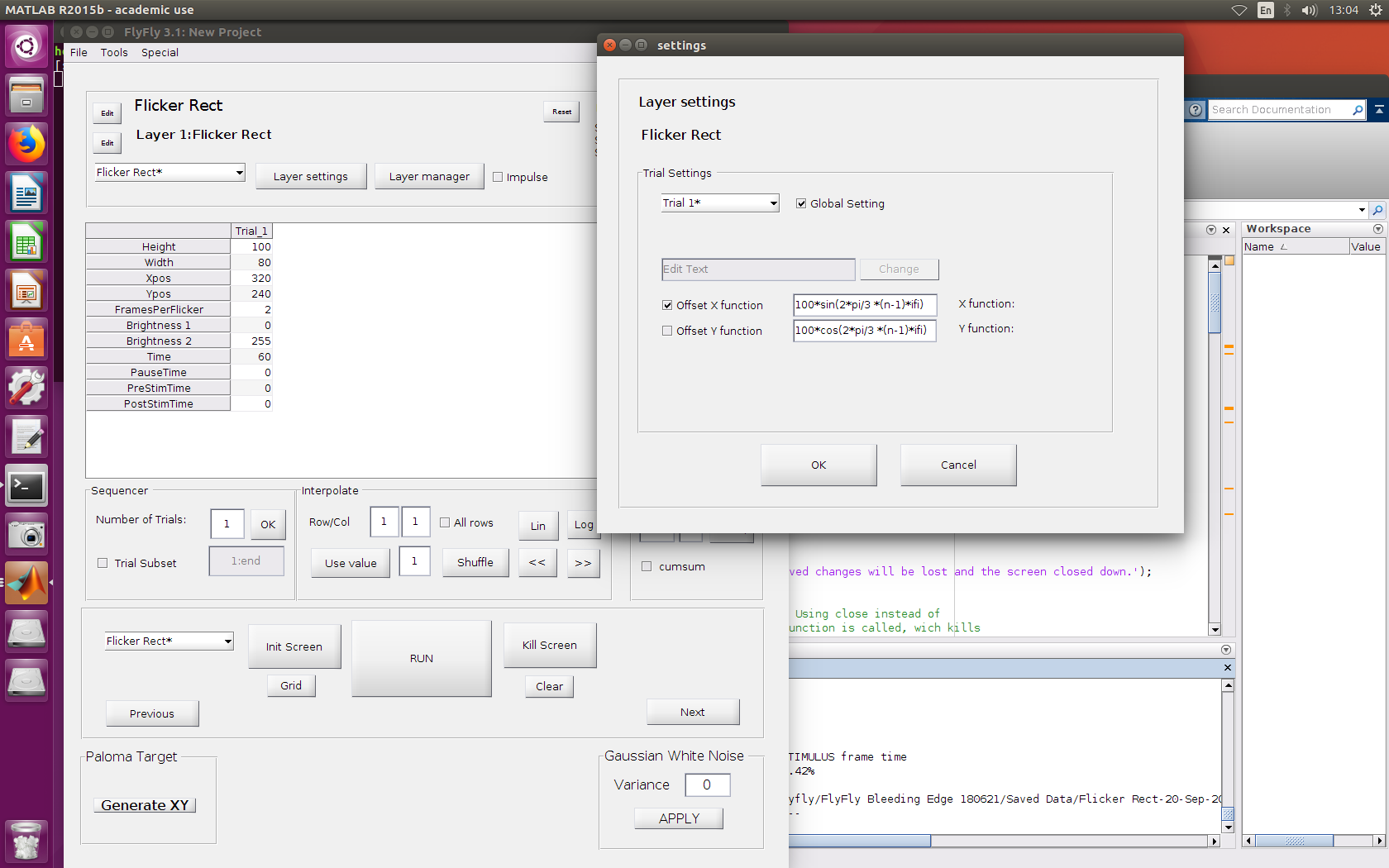
Unlike the stimulus settings for Aperture, in Dual Apertures the position Xpos = 0 and Ypos = 0 denotes the centre of the screen. Setting Xpos as -200 and Xpos2 as 200, puts an aperture on both sides of the screen at 200 pixels from the centre.

The default setting is for the apertures to be an ellipse, however this can be changed in *Layer settings* to a rectangle for either one or both apertures.

## FLICKER RECT

This stimulus displays a flickering rectangle that can be placed anywhere on the screen. The size of the rectangle is given by the *Height* and the *Width* (in pixels) and the location in *Xpos* and *Ypos* (in pixels, referring to the centre of the rectangle). The rectangle flickers between 2 brightness levels, which are given by *Brightness 1* and *Brightness 2.* Note that these are given on a scale from 0 (black) to 255 (white). The time each brightness level is displayed, before switching to the other brightness level, is given by *FramesPerFlicker*.

This stimulus has useful *Layer settings*, which allows for control of the movement of the flickering rectangle. By adding a function to the *Offset X* function or the *Offset Y* function the rectangle will move across the screen in the x, and y-plane, respectively. In the function, n refers to every frame. As you can see below, you can be very creative when describing the movements.

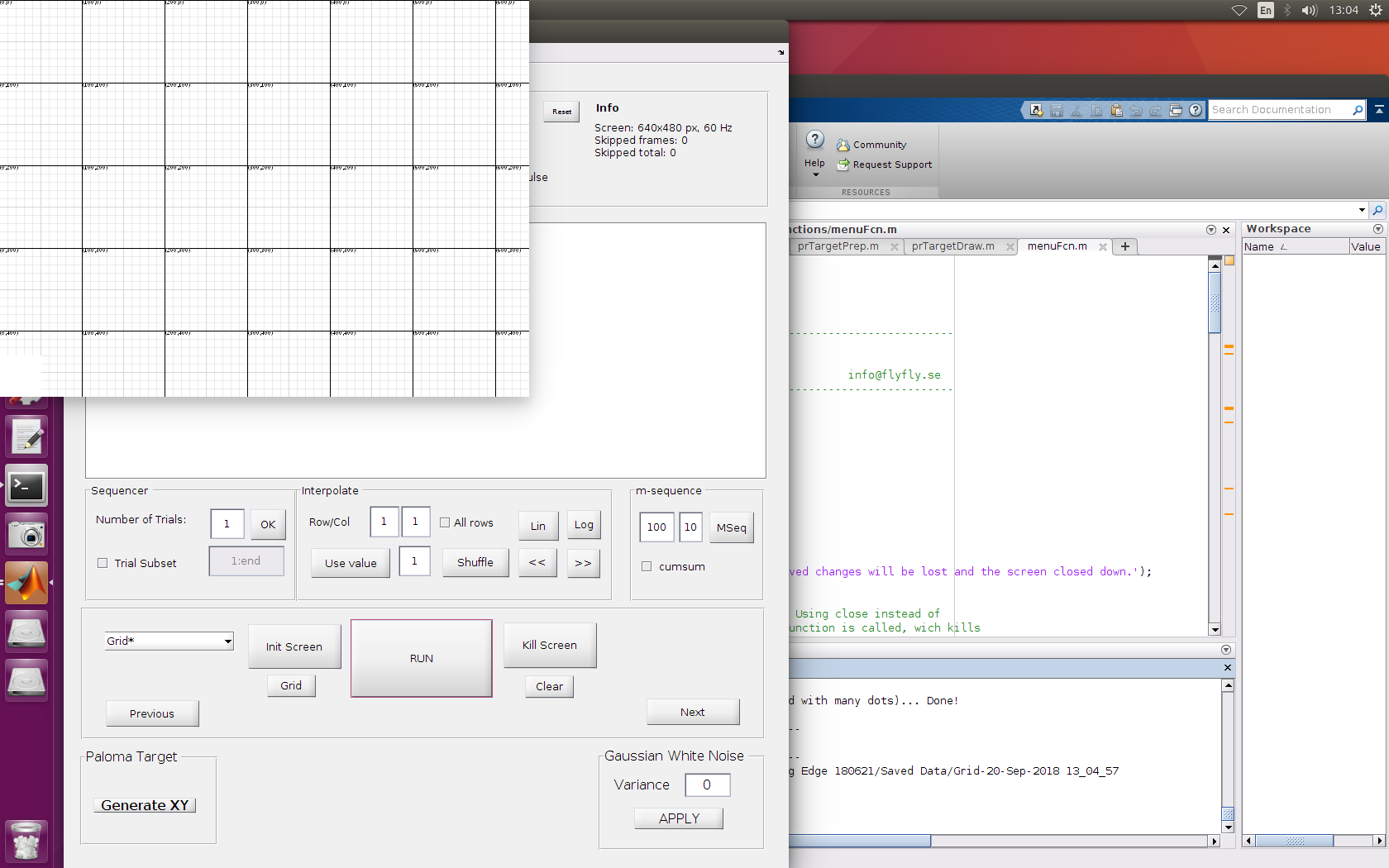
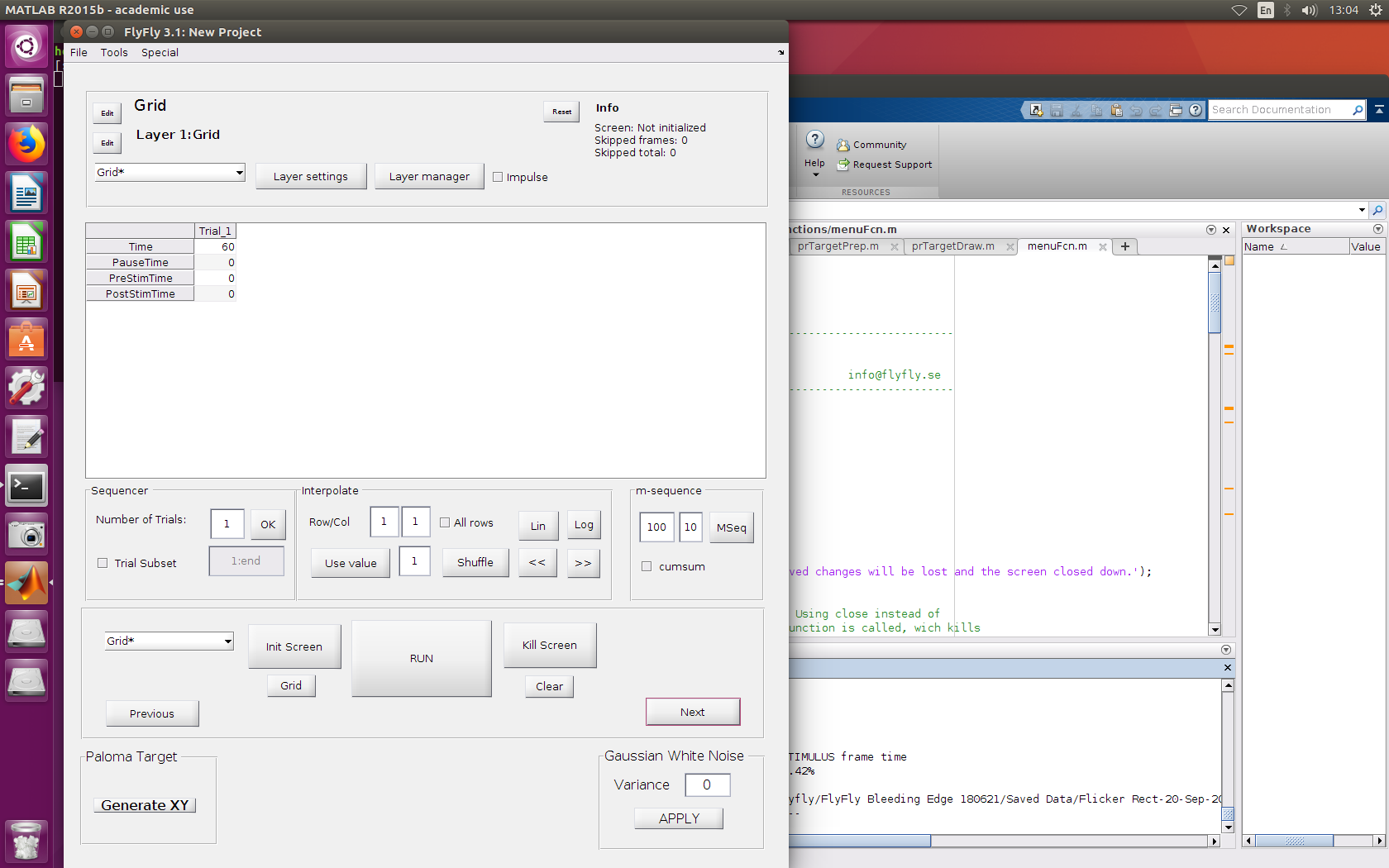


## MOUSE TARGET

This stimulus serves no practical purpose as is, and will not be of much use in an experiment. However, it is potentially very useful for *developers* of new FlyFly stimuli, as it provides an example of how to obtain and use mouse and keyboard input. Hence, it has been included in the list of available stimuli.

## GRID

The grid stimulus can be used to display a grid with the pixel values on the screen. This can be very useful when troubleshooting, but probably has little value during actual experiments.



## .**MAT SEQUENCE**

**Note that unlike the other stimuli, .Mat sequence is incompatible with layers. This means that you cannot combine a .Mat sequence stimulus with an aperture overlay, for example.**

This stimulus is the most general one in FlyFly, in the sense that it allows any sequence of pixel patterns whatsoever to be displayed (as long as it is in grayscale); but this sequence needs to be created appropriately first, and the creation process occurs outside of FlyFly.

The data for the stimulus is stored as a .mat file, containing a single 3-dimensional MATLAB matrix. The name of the 3D matrix must be “out”. (The name of the file containing it can be anything.) The third dimension of the matrix defines time, and the first and second dimensions define the Y and X dimensions of each frame, in pixels. So a “slice” through the matrix at one index of the third dimension,

out(:, :, n)

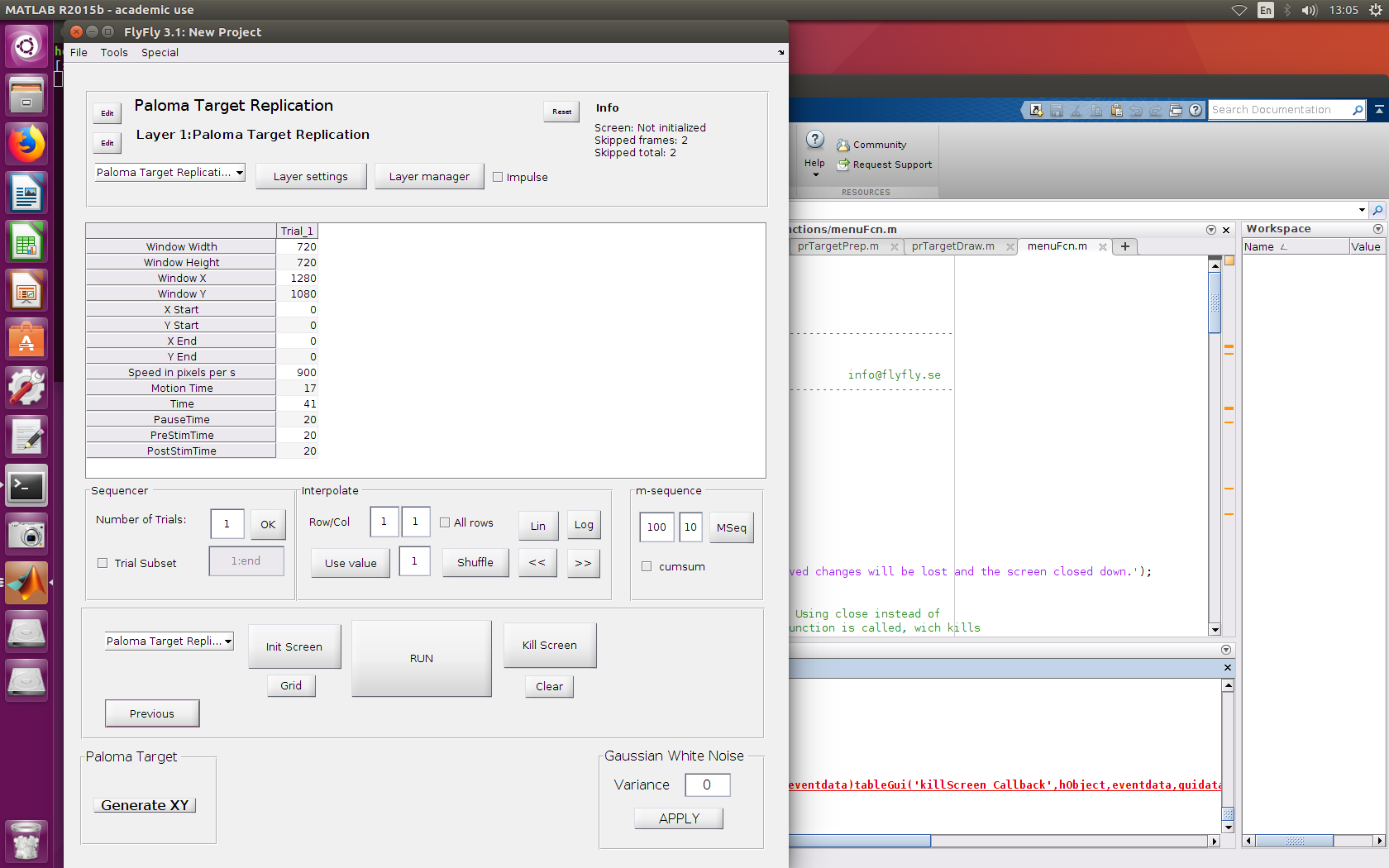
represents a single image to be displayed on the screen. The data values range between 0.0 and 1.0, and represent the colour (shade of gray) to be displayed at each pixel.

One particular “use case” for the .Mat Sequence is that it can be used to display a movie. The script “convert” (in the “Stimuli/MatSequence” subfolder) may be useful in helping to achieve this, although it has only been tested on a few movie files. It takes in two arguments, “infile” and “outfile”, converts the movie in “infile” into the matrix format that is required for this stimulus, and saves the matrix to “outfile”. If the movie is in colour, it is converted to black and white.

NOTE: It really does not like larger videos. More like, in this format, video file sizes get extremely big extremely quickly. For example a 1000x1000x825 (5s long video, not exactly a super long video) is > 100mb in size, and flyfly pauses for about 30 seconds after hitting run before playing the stimulus, while it loads it.

Interestingly there is the option to go fullscreen, which just expands your video to the full dimensions of the screen. Might be worth looking at that one day, so we can load smaller videos (XY size wise) and then expand them in FlyFly to reduce the file size and load time

## PALOMA TARGET REPLICATION



This stimulus was created to map the receptive fields of target sensitive neurons in a similar way to what is done in Paloma Gonzalez-Bellido’s lab. It’s a pretty specific stimulus, and for it to work you have to do things in a given order. The stimulus displays a rectangular target (size given, as 15x15 pixels, can only be changed in the code itself) appearing in a random location within a region of interest (*Window*), and moving in a random direction. The *Window Width* and *Window Height* are given in pixels, and the *Window X* and *Window Y* specifies the center location of the window on the screen, in pixels.

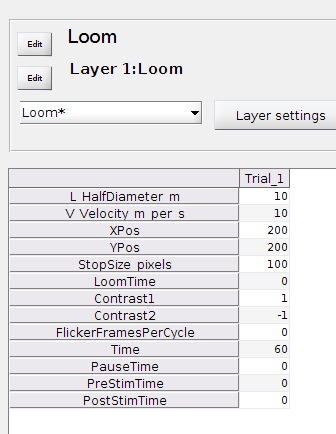
**Do not touch *X Start, Y Start, X End, Y End*!**

*Speed* in pixels/s.

The *Time* specifies the number of frames the target is displayed on the screen. The target starts off being stationary, and then it moves for the time given by *Motion Time*. This means that in the given example on the left, the target will be stationary for 24 frames (41 - 17) and then move for 17 frames.

Once the user has specified the number of trials, the size and location of the window, and the screen is initialized, press *Generate XY*. This will put a whole bunch of numbers into *X Start, Y Start, X End, Y End, and you’re ready to run your experiment.*

## LOOM



This stimulus generates a circular looming stimulus, using the equations by Gabbiani et al, J Neurosci, 1999. Briefly, it simulates the 2D projection of a circular with *radius L* (in m) object approaching the viewer at a constant *velocity V* (in m/s). The user specifies the *StopSize Pixels* (which refers to the diameter), and the *LoomTime* (in frames). These four values are used to calculate the start size and the growth rate.

The *XPos* and *YPos* specify where the looming stimulus should be centered on the screen.

The stimulus can flicker between two grey levels by using *Contrast1* and *Contrast2*, on a scale from -1 to 1. The flicker follows a sinusoidal pattern witty the wavelength given by *FlickerFramesPerCycle*.

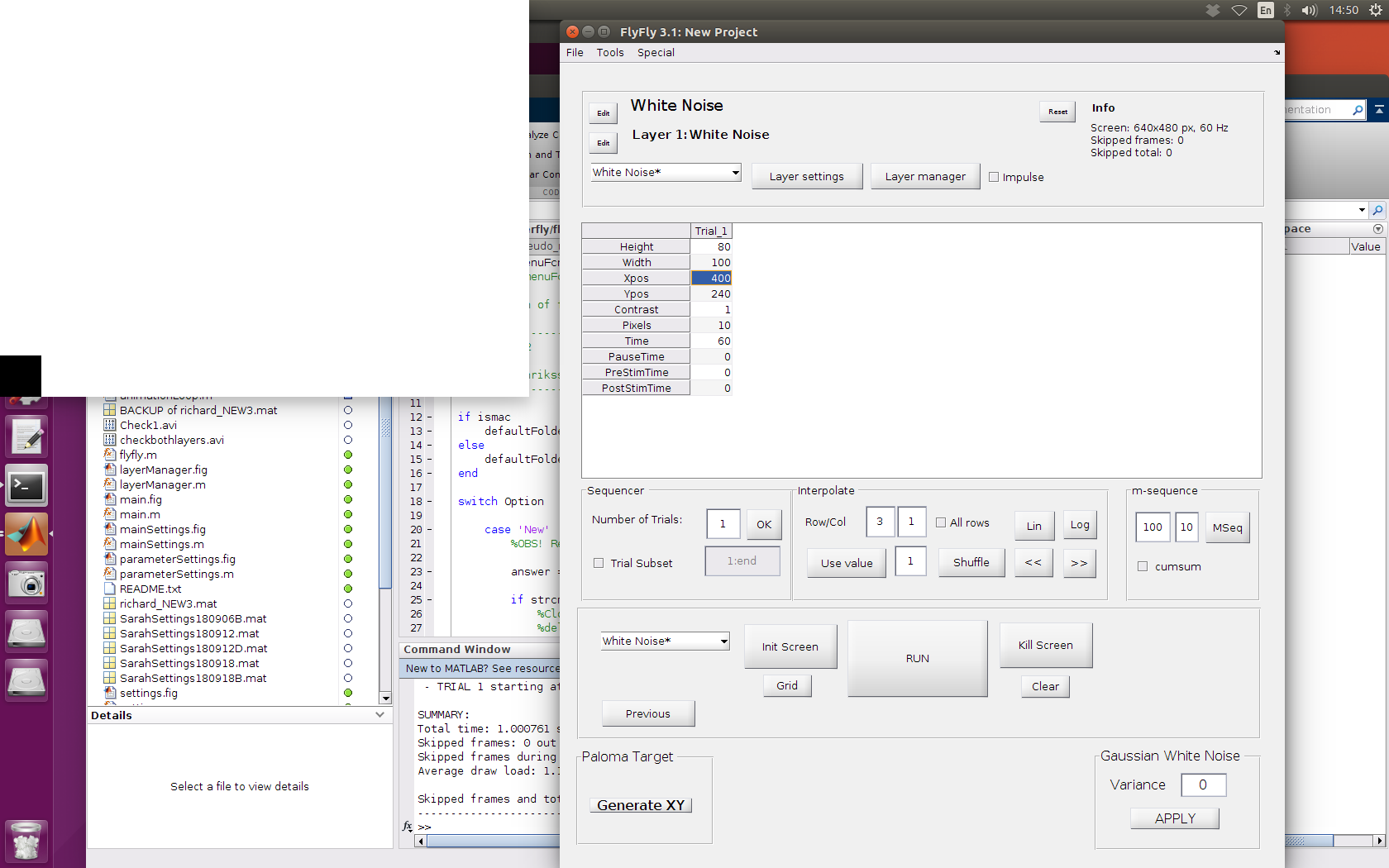
Note that if *Time* is longer than *LoomTime*, the stimulus is shown at the *StopSize* for the remainder of the time. For example, if *Time* is 120 frames, and *LoomTime* is 60 frames, the stimulus remains as a stationary circle with a diameter of *StopSize* for the last 60 frames.

**IMAGE TARGET FLICKER  
TARGET3D XYZ**

**RECT TARGET RELATIVE**

**STARFIELD FLICKER**

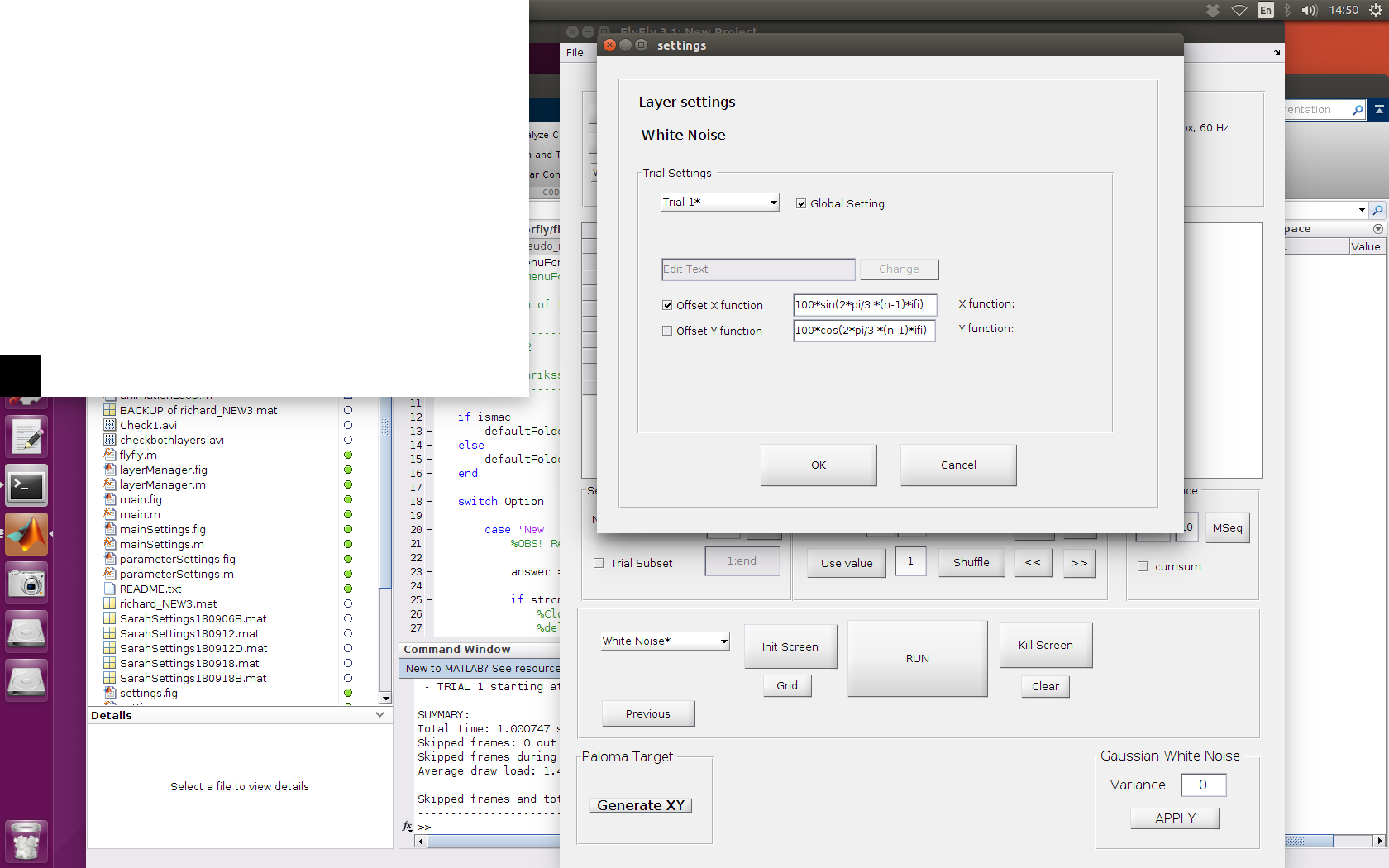
## WHITE NOISE



Gone from version 3.2?

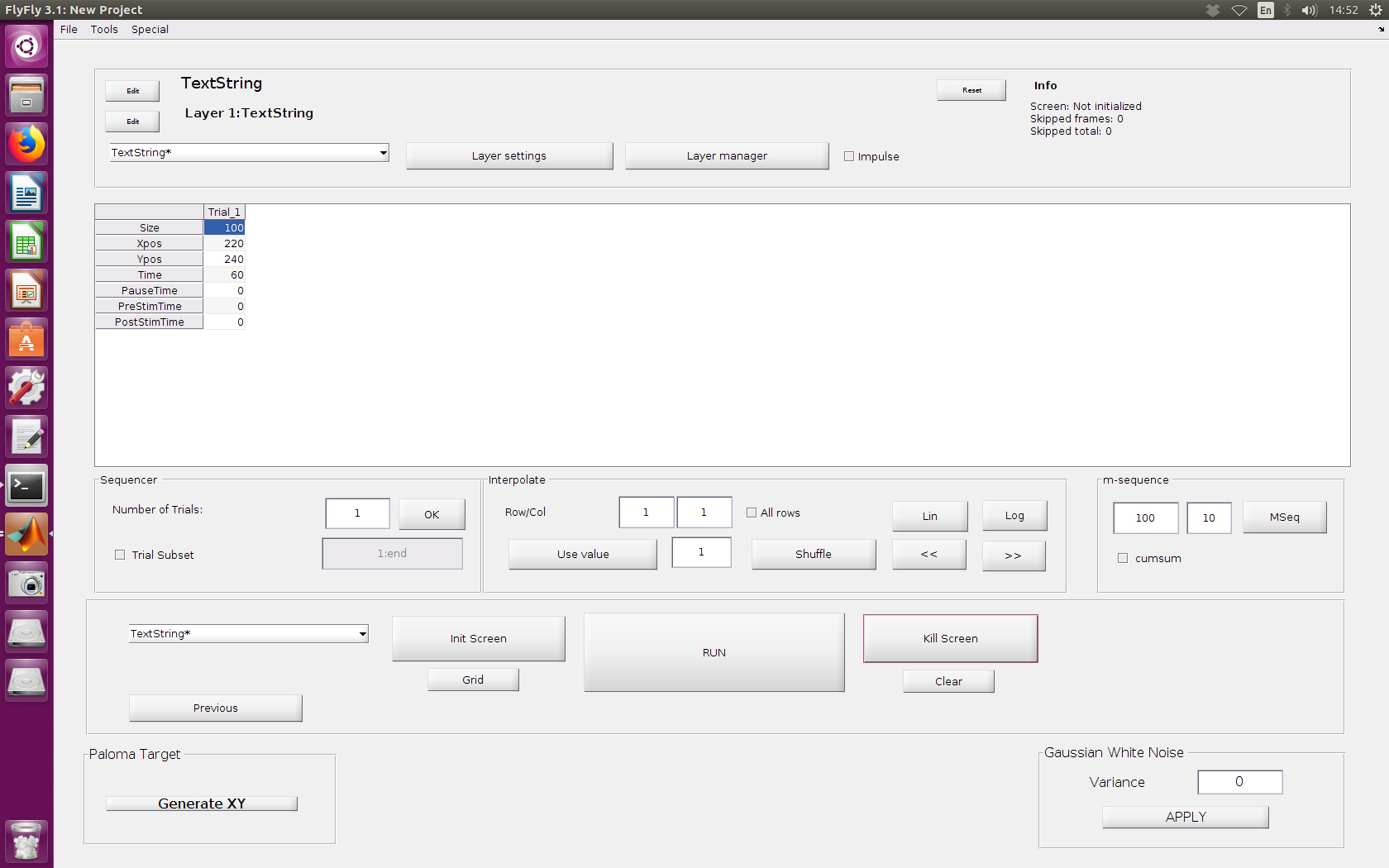
This stimulus displays a rectangle filled with white noise that can be placed anywhere on the screen. The size of the rectangle is given by the *Height* and the *Width* (in pixels) and the location in *Xpos* and *Ypos* (in pixels, referring to the centre of the rectangle). The Contrast is given on a scale from 0 (all pixels are grey) to 1 (all pixels are black and white). The size of each flickering component can be given by Pixels

This stimulus has useful *Layer settings*, which allows for control of the movement of the flickering rectangle. By adding a function to the *Offset X function* or the *Offset Y* *function* the rectangle will move across the screen in the x, and y-plane, respectively. In the function, *n* refers to every frame. As you can see below, you can be very creative when describing the movements.



## TEXT STRING

Gone from version 3.2?



This stimulus can be used to show a text string on the screen. The size and location are given by the parameters, whereas the actual text is given in the *Layer settings*.

## Parameter saving structure

Before FlyFly executes a stimulus, it saves all parameters related to that stimulus to disk.

The file is automatically named so as to uniquely identify the particular stimulus. The file name has the structure

<Stimulus Name>-<Date> <Starting Time>.mat,

Where <Starting Time> is equal to the parameter timeStart described below.

The parameters are saved as a .mat file, containing the following Matlab variables:

* timeStart - string representing the time at the start of the *saving of the parameter file* (system timestamp from the FlyFly machine), to the nearest second (Note: this is *not* the time of the start of stimulus presentation!)
* timeStartPrecision - the same as timeStart, but represented as a length-6 array with date broken down into year, month, day, hour, minute, seconds (to subsecond precision)
* timeFinished - string representing the time when the stimulus ended
* message - string: either 'NOTE: STIMULUS PLAYED TO THE END', when the stimulus was successfully completed, or 'NOTE: THIS RUN WAS ABORTED' when the stimulus was terminated by FlyFly for any reason. (Parameters are saved before and after the stimulus is run. The “aborted” message is saved before running, and the “to the end” message is save after running.)
* stimulus - the core details of the actual stimulus that was specified(discussed in more detail below)
* debugData - additional data describing the stimulus and the way it executed (discussed in more detail below)

### stimulus

The stimulus struct contains two fields: name, which is the name of the stimulus, and layers, which is a struct array describing each of the layers that the stimulus consists of. Each item in layers has the following structure:

* name - the name of the layer
* fcnPrep, fcnDraw - the names of the Matlab functions used to create and render the stimulus
* impulse - whether the stimulus was run in impulse mode or not
* Param - a struct array of length equal to the number of trials, containing the actual parameter values specified for each trial
* settings - a struct array of the same length as Param, containing the corresponding Layer Settings for each trial.

### debugData

This variable contains mostly information that is useful for developers of FlyFly or FlyFly stimuli. However, it does also contain information that can be used to examine frame timing, including the numbers of frames dropped during stimulus presentation. It contains the following variables:

* stimulus - contains the same information as the stimulus variable described above. The format of this one is probably more difficult to work with.
* screenData - information about the properties of the screen
* userSettings - miscellaneous settings used by FlyFly
* trialSubset - a list of the trials that were selected to run
* dataLog - information about frame timing (discussed in more detail below)

The dataLog is a struct array containing timing information for every frame that was scheduled to run (whether it was actually displayed, or was skipped/dropped). The size of dataLog is TxF, where T = the number of trials, and F = the maximum number of frames scheduled in any of the trials. Each item in dataLog corresponds to a single frame in each trial, and has the following structure:

* frameDelay - zero if the frame was not dropped; if non-zero, the value is the number of frames that have been dropped (Note: this is only a rough estimate. Nevertheless if frameDelay is zero, it is safe to assume no frames were dropped.)
* missed - a value returned by Screen(Flip) in PsychToolbox. A negative value indicates that the deadline for displaying the frame was met, while a positive value indicates that the deadline was missed (and hence that a number of frames were dropped)
* time - the amount of time used in rendering the frame (or attempting to render it, if frames were dropped)
* frames - a Lx1 array, where L = (number of layers + 1). The value of frames for each layer is the number (in order) of the frame for that trial, for that layer. The last element of frames gives the trigger value displayed for the photodiode trigger.

### Note on the interpretation of timeStart

It is convenient (and in many cases it may be accurate enough) to treat timeStart as if it represents the onset time of the stimulus, but as described above, this is not really correct. The value of timeStart is determined just before the parameter file is saved (with the stimulus commencing immediately after saving). Hence the value of timeStart is *always* earlier than the actual stimulus onset time, and deviates from it by the amount of time required to save the parameters file. The size of this deviation will be system-specific, and will increase according to the number of trials and the number of layers. Also keep in mind that it is possible for a particular kind of stimulus to include arbitrary additional information, as determined by the developer of the stimulus, into the parameters file, so that the time deviation is not predictable in principle. Even when the number of trials is small and no additional information is included, users of FlyFly and analysts of the data should treat timeStart as an approximation of the stimulus start time, and should not rely on it for high precision.

For example, when performing data merging using FlyFlyDataMerger, the user is required to provide a time offset between “data block time” and “FlyFly time”, where the latter is taken from timeStart. When one stimulus in a recording has a large number of trials (on the order of hundreds), this time offset will not be constant, but will vary substantially over the course of a recording, due to variations in the numbers of trials in various stimuli, and in such cases it may not be possible to merge all data from the recording in a single pass with DataMerger. The workaround in this case is to separate out the longer stimuli and merge them separately.

1. Note that there are limitations to the way that the Starfield is rendered. There are no cues to 3-dimensional shape, such as lighting, to support interpretation of the circles as spheres. Also, the size of the dot is the spherical diameter of the underlying sphere, not the actual perimeter of a 3-dimensional sphere that would be visible from the fly’s position; see Stott et al. (2018), J Exp Biol 221, for a discussion of this point.)

   A more fundamental limitation applies to all FlyFly stimuli: the view is projected onto a 2-D screen in front of the fly, rather than as an array surrounding the fly. This projection assumes a viewer with only a single eye, with a line of sight perpendicular to the screen. [↑](#footnote-ref-0)