FlyFly 3.1 User Manual

# Overview

FlyFly is a software system to set up and manage experiments that involve the display of visual stimuli on the computer screen. It runs on top of the MATLAB programming environment.

Researchers can assemble experiments by selecting visual stimuli to be displayed in sequence, either from stimuli that are provided as part of the system, or from FlyFly stimuli that have been created by the researchers themselves. It is possible to set various parameters to control how the stimuli are displayed, as well as the timing of presentation.

Once experiments have been set up, they can be “run”, and the stimuli will be displayed on the screen that the user has specified. FlyFly makes use of the Psychophysics Toolbox library to display stimuli on-screen.

FlyFly was designed for use in experiments on the visual system of insects, but as the software allows the creation of custom stimuli, it should in principle be possible to use it for other kinds of experiments too.

# New in FlyFly 3.0

The major changes in FlyFly 3.0 compared to previous versions are:

* Stimulus durations are now specified in terms of *number of frames*, rather than number of seconds, for greater repeatability of stimuli
* Additional capabilities for randomization have been added in the form of   
  *m-sequences*
* Two new stimuli, Rolling Image and Rolling Image MII, have been added

In addition, a new Starfield stimulus was created (as “Starfield\_New”), but not completely integrated into FlyFly. This stimulus has been extensively redeveloped in FlyFly 3.1 (as “Starfield 2”), and it is recommended that Starfield 2 in v3.1 should be used instead of Starfield\_New in v3.0.

## Stimulus Durations: Number of Frames

As of version 3.0, users should specify the duration of each stimulus (in the Time field) as the number of frames to be rendered, rather than as time measured in seconds. This allows for more precise and repeatable timing, as it is no longer necessary for the software to calculate the actual number of frames from time and refresh rate, which created the risk of inconsistent or unpredictable numbers of frames due to rounding.

# New in FlyFly 3.1

The major changes in FlyFly 3.1 are the addition of three new stimuli:

* 3D Target
* Starfield 2: 3D Space
* Image Target

See the rest of this document for details on all new features.

## Installation and Getting Started

## Required components

To run the program you need

* MatLab 10 or later (R2014b or later is recommended)
* Psychophysics Toolbox 3 (*Psychtoolbox-3* or PTB-3)[[1]](#footnote-1)
* A folder containing all the FlyFly files, unzipped from the downloadable zip file.

Older MatLab versions should work with some changes in the program code, such as replacing the tilde character[[2]](#footnote-2) (~) with a temporary variable. Older versions of PTB have not been tested and might not function. If you don´t have Psychophysics Toolbox you can get it at **psychtoolbox**.**org**/.

## Launching the program

Open Matlab and change the *current folder* to the root folder of FlyFly - this folder is named FlyFly 1.7.1 or similar. Launch the program by running the file flyfly.m (you may either right click on the file in the Current folder view and select Run file or just type flyfly in the command window).

## The main window

Running the file flyfly.m will start the *main* application. You should see a window similar to Figure 1 (with possible small deviations due to version history).

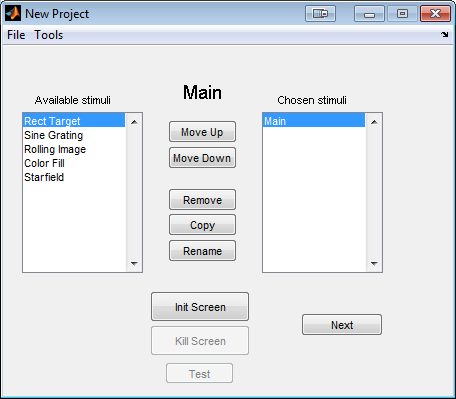


Figure 1: main view

To the left you see a list of available experiments. Double click one or several experiments to add them to your project, i.e. the list *Chosen stimuli*. You can add several experiments of the same kind, using different parameter settings. For example, with the *target* experiment, the first stimulus could be used to map a receptive field and the second to do a size tuning experiment.

Once added, it is possible to remove or rename the experiments. It is also possible to copy an experiment, which might be useful if you want to make small changes to an existing experiment.

For experienced users, it is possible to import parts from an existing project. In the File menu, the "Import Stimuli" option makes it possible to select an existing saved project and specify which experiments in that project to import.

Clicking the 'Next' button will send you to the first experiment. If you haven´t added one, nothing will happen.

## The screen

To run any experiment you will need to initialize something called a Screen. If you run a dual display setup you will probably use one monitor to control the user interface and one to display the stimuli. To see the numbering of your available monitors either type Screen('Screens') in the command window or press Tools -> Settings from the main view. Under the Screen panel there should be a line saying "Available screen numbers: " followed by the numbers (starting at 0). For a single monitor this will be '0', with two monitors '0, 1' etc.

From the settings panel you can change which screen to use (if you have multiple) and also switch between full screen or a smaller partial screen. The partial screen is by default placed with its left top corner in the left top corner of the monitor with the task bar. To change this position you have to make a change in the matlab file *initFcn.m*.

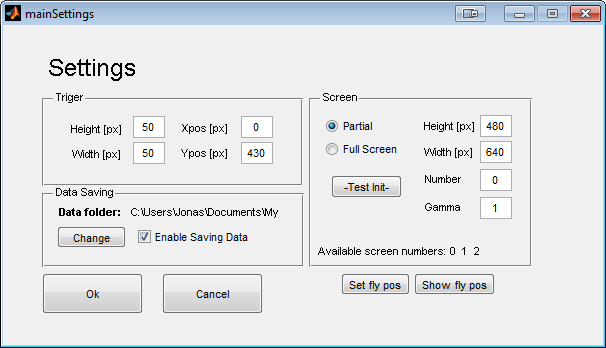


Figure 2: Main settings

The screen is initialized by clicking on the "Init Screen" button. Once the screen is initialized, it covers everything behind it, which makes it impossible to click there. You can close the screen by clicking the "Kill Screen" button or by typing Screen('CloseAll'); in the matlab command window.

**IMPORTANT**: If you cover both the matlab command window and the FlyFly GUI, it is impossible to close the screen in a neat way. In OSX you can force quit matlab by pressing *cmd+q*. In windows it is usually possible to get the taskbar in front of the screen using alt+tab, which allows the screen to be closed (right click -> close window) .

## Settings

Pressing Tools -> Settings opens up the settings panel.

### Trigger

From the settings panel it is possible to change the size and position of the photodiode trigger. The coordinate system used for the position starts in the top left corner with positive x to the right and positive y downwards.

The trigger color is by default white (on) and black (off). To change it, find the variables triggerRGBon and triggerRGBoff in the file *initFcn* and change the values accordingly. Note that you have to restart the program to apply these changes, because the file runs at start-up.

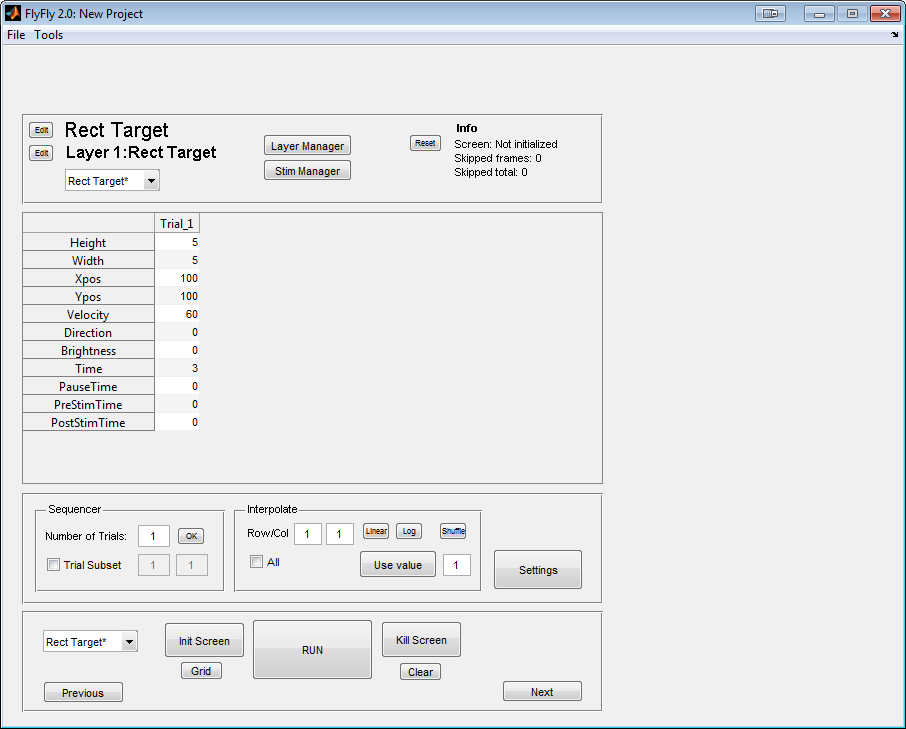
### Screen

It is possible to change the gamma of the screen used.

The button called Init Test . Pressing this button will initialize a new window and then automatically close it after 2s. If you´re unsure which monitor corresponds to which number, use this function in order to test, so you don´t cover the user interface by accident.

## The tableGui

This Gui lets you enter parameters from a table view, which is suitable for many types of experiments.



1

6

5

4

3

2

**1:** Name of experiment (*Rect* *Targets*) and current layer (*Layer 1: Rect Target*). You can change the names by pressing the *Edit* buttons on the left. You can add more layers in the layer manager.

**2:** Info panel. Gives the resolution and update frequency of the screen, if it has been initialized. *Skipped frames* gives the number of skipped frames for the last experiment and *Skipped total* gives the total number of frames skipped. This value can be reset with the *Reset* button to the left.

**3:** The parameter table. Here it is possible to set all the parameters to be used in the current layer. If a cell is marked in the table you have to click once outside the table to be able to make a new action, such as clicking on a button.

**4:** Sequencer panel. Change *Number of trials* to run by typing in a new number and pressing *OK*. Marking *Trial Subset* will play the trial(s) between the chosen numbers.

**5:** Interpolate panel. The *Linear* and *Log* buttons interpolate between the end values (first and last) on a single row in the parameters table. The row is chosen by clicking on the row in the table. The shuffle button shuffles the order of values in the row randomly.

The *Use value* button takes the currently selected value and applies it to all the *n*th cells in the same row, where *n* is the number in the box to the right of the button (not shown here) . I.e. if the number in the box is 2, then every second value will be the same as the marked one.

Checking the 'All' box means the same action will be applied to all rows at the same time.

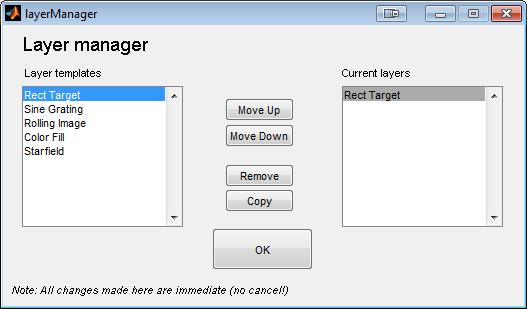
**6:** Before running an experiment the screen needs to be initialized by pressing *Init Screen*. At initializing, a number of tests is run by PTB-3 to measure the screen update frequency and synchronize with the vertical retrace. Sometimes, notably on computers with older graphic cards, this fails and an error message will be thrown. In that case just press the button again.

To close down the screen after an experiment press the button *Kill Screen*. The screen will also automatically be closed when the program itself is shut down.

You can draw a grid to the screen by pressing the *grid* button. This is useful to get a quick idea on which coordinates to use in your experiment. You can clear the screen again by pressing the *clear* button.

## 

## Layers and the Layer Manager



All stimuli are implemented as layers. This means that all stimuli can be combined together if needed. From the layer manager it is possible to add new layers from the list of available stimuli. Stimuli added will draw "over" the previous one, meaning that if you want for example a small target moving over a sine grating, the target should be added last or moved to the bottom of the list to be visible.

You need at least one layer to draw a stimulus.

## Implemented Stimuli

## The 'target' stimulus

//Parameters

'Height' - Target height in pixels

'Width' - Target width in pixels

'Xpos' - Target starting position in pixels

'Ypos' - target starting position in pixels (top is 0)

'Velocity' - Target speed in pixels/s

'Direction' - Target motion direction

'Target Brightness' - Target brightness (0-255, 0=black, 255=white)

'Background' - BG brightness (0-255, 0=black, 255=white)

'Time' - How many frames the target will be rendered

'PauseTime' - Length of pause between trials. During pause the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

The *target* stimulus draws a black square to the screen and moves it at a uniform speed for a specified amount of time.

## The 'sine' stimulus

//Parameters  
'Wavelength' - Wavelength in pixels

'Temporal Freq' - Temporal frequency (cycles/s)

'Direction' - Direction of motion

'PatchHeight' - Height of grating patch in pixels

'patchWidth' - Width of grating patch in pixels

'Patch Xpos' - Patch position, left

'Patch Ypos' - Patch position, top

'Contrast' - Patch contrast (0-1: 0 = gray image, 1 = normal)

'Time' - How many frames the grating will be rendered

'PauseTime' - Length of pause between trials. During pause the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

The *sine* stimulus draws a patch filled with a sinusoidal grating.

## Rolling Image and Rolling Image MII

The Rolling Image stimuli display an image on the screen and moves it inside a defined image window. The window itself can also move across the screen at the same time. For example, one option for movement internal to the window is to make the image loop around within the window indefinitely, scrolling from left to right (with the left and right edges attached to each other).

The “original” Rolling Image stimulus is no longer actively supported, but is being used in some experiments and so will not be deleted. For new experiments, Rolling Image MII supersedes Rolling Image, as it is possible to recreate all Rolling Image stimuli using Rolling Image MII.

Movement is set inside the layer settings of Rolling Image MII, by specifying snippets of Matlab code that are executed at each point in the animation loop in order to calculate the relative offset of the image to the image window (using ‘Image offset’), as well as the location of the window relative to the screen (‘X pos = …’ and ‘Y pos = …’).

The Matlab code will be executed inside the drawing routine (rolling ImageM2Draw) at run-time, and hence the user has access to all variables available in that particular context. It is expected that the only variable most users will be interested in is n, the index of the frame currently being rendered. But in principle, it is of course possible to use any of the other variables defined at this point, or even define your own variables. (Keep in mind that the last statement to be executed should return the value to be used as the offset, and so should NOT be terminated with a semi-colon!). It is also possible to insert any arbitrary snippet of Matlab code (including deleterious code) at this point, so please be careful if you insist on trying out anything ‘weird’.

A more ‘normal’ use of the ‘Image offset’ function would be simply to enter ‘n’, which would scroll the image from left to right.

The ‘Image offset’ setting has a default function 100\*abs(2\*(n/1 - floor(n/1+0.5))). This is a specific version of the more general function C\*abs(2\*( k1n- floor(k2n+p))), and the values of the four parameters can be manipulated to create various kinds of movement. The Image Offset function governs the movement of the complete image within the image window created on the screen (for instance, by scrolling a larger image from left to right within a smaller window).

The “X Pos” and “Y Pos” settings govern the movement of the window itself. The default X Pos function is 100\*sin(n). Reducing the frequency of this oscillation (for example 100\*sin(0.1\*n) ), produces a regular oscillation that moves less rapidly.

Rolling Image MII

//Parameters

‘Direction' - Angle of rotation of the image (in degrees)

(NOTE: only the range -180 to 90 is supported!)

'Xpos' - X position of the middle of the *image window*

'Ypos' - Y position of the middle of the *image window*

'Height' - Height of image window (crops image if smaller

than image height)

'Width' - Width of image window (crops image if smaller

than image width)

'Contrast' - Scale factor for contrast to be applied to image

(leave equal to 1.0 to retain original contrast)

'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause

the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

//Stimulus Settings

'Xpos' - X position of the middle of the *image window*

'Ypos' - Y position of the middle of the *image window*

'Height' - Height of image window (crops image if smaller

than image height)

'Width' - Width of image window (crops image if smaller

than image width)

'Contrast' - Scale factor for contrast to be applied to image

(leave equal to 1.0 to retain original contrast)

'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause

the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

## The original 'rolling image' stimulus

## //Parameters 'Speed' - Speed (pixels/s)

'Direction' - Angle relative to monitor top

'Xpos' - Image offset on x-axis

'Ypos' - Image offset on y-axis

'Height' - Image height (pixels)

'Width' - Image width (pixels)

'Starting position' - Starting position of image

'Contrast' - Contrast between 0 and 1

'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

Sets a background image and moves it across the screen. The image will have its top left corner in the corner of the screen. Height and width is size of visible patch of image. Making the patch larger than the image will not stretch it. *Xpos* and *Ypos* move the patch relative to its starting position.

## Starfield 1

//Parameters  
'Brightness' - Brightness of dots  
'Dot size' - Size of dots when right in front of the viewer  
'number of dots' - Number of dots used in calculations  
'Sideslip' - Sideslip (cm/s)  
'Lift' - Lift (cm/s)  
'Thrust' - Thrust (cm/s)  
'Pitch' - Pitch (deg/s)  
'Yaw' - Yaw (deg/s)  
'Roll' - Roll (deg/s)  
'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

Draws a cylinder of dots and translates and or rotates it. Number of dots is the number of dots in the cloud, which is not the same as number of dots on screen. Dots on screen depend on the total number of dots, distance to monitor and monitor width (adjusted in settings).

## Starfield 2: 3D Space

This stimulus was based on the (unused) stimulus “Starfield\_new” in FlyFly 3.0. The stimulus creates a 3-dimensional space of spherical objects (the “starfield”) at various distances from the screen. These objects appear as black circles on the screen, becoming darker in colour and larger the closer they are to the screen.

The purpose of Starfield 2 is generally to simulate a cluttered background containing several objects. The objects in the background move together simultaneously. The starfield can translate in the X, Y and Z directions, or rotate around the X, Y or Z axes. NOTE that only one of these movement transformations should be applied per trial. Compound rotations, or combined rotations and translations, are not supported yet (although compound translations will work correctly).

One use case for Starfield 2 is to use it in conjunction with 3D Target (see above), to simulate a target moving over a cluttered background. Several trials can be placed in sequence to define a continuous 3D trajectory for the target. In this scenario, it would be necessary to set the “Retain into next trial” parameter in Starfield to 1, to ensure that the starfield persists across trials.

//Parameters

'Dot size' - Size of dots when right in front of the viewer  
'Dot density’ - Number of dots per 10,000 cubic centimetres  
'Sideslip' - Sideslip (cm/s)  
'Lift' - Lift (cm/s)  
'Thrust' - Thrust (cm/s)  
'Pitch' - Pitch (deg/s)  
'Yaw' - Yaw (deg/s)  
'Roll' - Roll (deg/s)

'Background Noise' - Proportion of random noise to be applied to the

starfield movement in each step. Values greater

than 1 may result in small movement increments in

the opposite direction from the specified

movement.

'Retain into next trial'- If set to 1, the next trial will use the same

starfield as the current trial, and the starfield

will be located at the same starting position

relative to the viewer at which the current trial

ends. If set to 0, a new starfield will be

generated for the next trial.

'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause

the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

## 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. Every other stimulus uses 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 3D Target, and can be edited, but supplying a value here will have *no* effect.

//Parameters

‘Target Size’ - Size of target when directly in front of viewer

(not in plane of screen!), in pixels

'Target Start Azimuth' - Azimuth angle of start position (in degrees)

'Target Start Elevation'- Elevation angle of start position (in degrees)

'Target Start Distance' - Straight-line distance between viewer and

start position (in cm)

'Target End Azimuth' - Azimuth angle of end position (in degrees)

'Target End Elevation' - Elevation angle of end position (in degrees)

'Target End Distance' - Straight-line distance between viewer and

end position (in cm)

'Velocity' - Target speed in pixels/s

'Target Noise' - Random noise to be applied to the movement of the

target

'Time' - How many frames the image will be rendered  
 (**NOTE**: THIS FIELD IS **IGNORED** IN 3D TARGET!

Instead, Time is calculated from Velocity)

'PauseTime' - Length of pause between trials. During pause

the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

## Image Target

This stimulus is an image counterpart to “Rect Target” (alternatively, a moving, rather than rolling, version of Rolling Image). It displays an image moving through 2-dimensional space (the plane of the screen), in a straight line and at a specified speed, duration and direction. The intention is to simulate a 2D target that can be any shape, rather than just a black rectangle as with “Rect Target”.

//Parameters

'Height' - Target height in pixels

'Width' - Target width in pixels

'Xpos' - Target starting position in pixels

'Ypos' - target starting position in pixels (top is 0)

'Velocity' - Target speed in pixels/s

'Direction' - Target motion direction

'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause

the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

// Layer Settings

'Image Path' - Location of the image file

## The '.Mat sequence'

//Parameters   
'Fps' - Frames per s  
'Xpos' - Placement on screen (center)  
'Ypos' - Placement on screen (center)  
'Time' - How many frames the image will be rendered

'PauseTime' - Length of pause between trials. During pause   
the trigger is turned off.

'PreStimTime' - Length of pre stim

'PostStimTime' - Length of post stim

Loads a 3D matrix and displays each "layer" (z-dimension) of it as a frame. Note that this takes up quite a lot of memory, meaning large matrices and/or long sequences might be impossible to play.

## 

## Run-time performance

There are a number of things that can be done in order to improve the general performance of the system running the stimuli. The most important thing is to make sure the operating system used is as clean as possible with a minimum of programs running. If possible all automatic updates, virus scans and similar should be turned off. It is also beneficial to disconnect the computer from the network and to unplug external devices. Depending on the OS, there might be a number of performance enhancing settings to do, such as turning off unnecessary graphical bling-bling.

## Common problems

*How can I abort a sequence of trials currently running?*

-Press ctrl-c in the matlab command window. Note that quiting like this means that matlab never reaches the end of the function running, meaning **no parameters will be saved**. Also, since the experiment is not finished properly it is not possible to start a new one (see below).

*The "Run button" is grey, even though I initialized the screen.*

-Try pressing the "Clear" button (will update screen status and refresh the GUI).

*The screen won´t initialize!*

-Try switching between "partial" and "fullscreen" mode. Make sure a valid screen number is entered.

1. An earlier version of the User’s Guide (for FlyFly 2) warned that Psychophysics Toolbox was incompatible with 64-bit Matlab. This appears to be no longer the case for the latest version of Psychtoolbox at the time of writing (PTB3.0.13), and in fact future releases of PTB will support only 64-bit Matlab. Consult the Psychtoolbox website for further details on version compatibility for your operating system. [↑](#footnote-ref-1)
2. Used to ignore the output from a function, first available in MatLab 10 [↑](#footnote-ref-2)