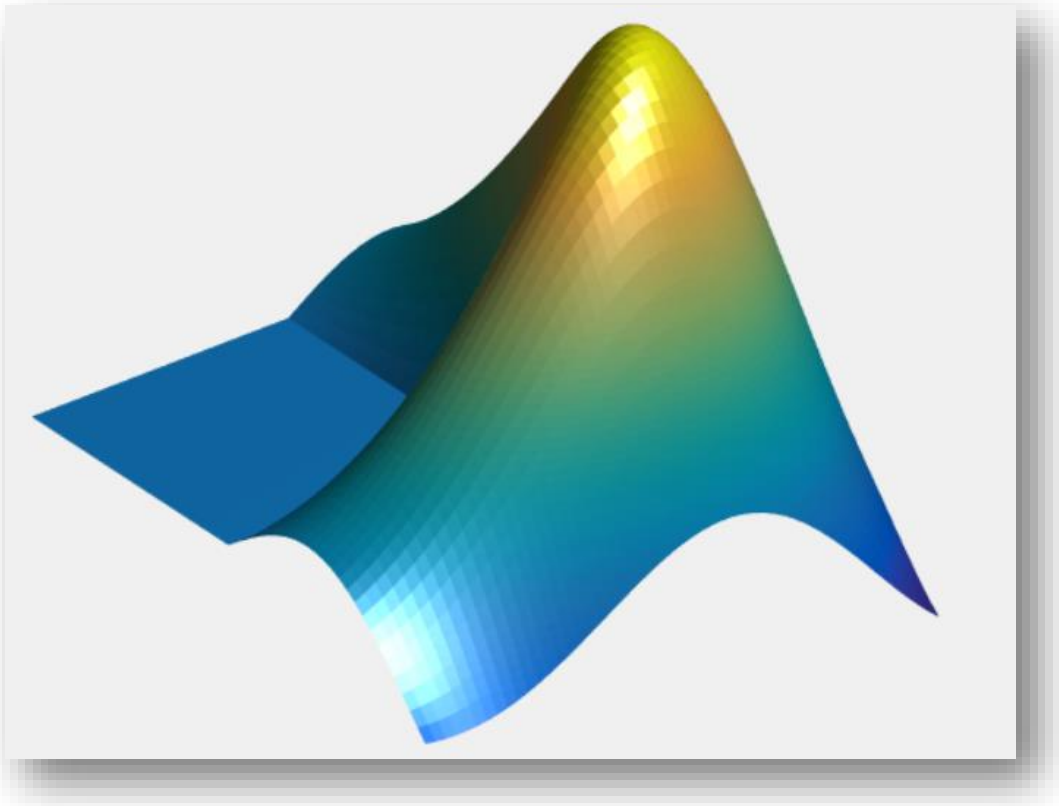
PsychToolbox: Part I



NSC 546 Computing for Neuroscience

Psychophysics Toolbox

Free set of Matlab and GNU/Octave functions developed originally for vision research, but now also used to support experiments in other areas of psychology and neuroscience.

Matlab functions provide precise control over video display properties. It provides routines for precisely timed presentation of images or videos and precisely timed registration of subject responses.

Installation

http://psychtoolbox.org/PsychtoolboxDownload

Single Monitor vs. Dual Monitor Setup

Programming in Psychophysics Toolbox can be easier if you have a dual monitor setup (but this may not always work right).

You can program and debug in Matlab on one monitor and view your experiment on the other monitor at the same time.

PsychToolbox is Working

```
>> ScreenTest
```

e.g., can check that dual monitor setup will work

Help

```
% get general help
help PsychToolbox

% find demos
help PsychDemos

% basic PsychToolbox command
help PsychBasic
```

Screen() command

probably the most basic of PsychToolbox commands

```
% Mac:
% Nscreen 0       main monitor that has Matlab running
% Nscreen 1       external monitor
%
% PC:
% Nscreen 0      "virtual display" (main and next)
% Nscreen 1      real main monitor
% Nscreen 2      external monitor

Nscreen = 0;

% basic OpenWindow command
[wPtr, rect] = Screen('OpenWindow', Nscreen);
```

Command Help

```
% help on the Screen command

% documentation on the Screen command
help Screen

% see Screen usage
Screen

% see help on a particular Screen call
Screen OpenWindow?
```

Screen() command

```
[wPtr,rect]=Screen('OpenWindow',
windowPtrOrScreenNumber [,color] [,rect]
[,pixelSize][,numberOfBuffers][,stereomode]
[,multisample][,imagingmode][,specialFlags]);
```

Note: parameters surrounded by [ ] are optional

Screen() command

```
[wPtr, rect] = Screen('OpenWindow', Nscreen);
```

Passed Parameters:

`'OpenWindow'`	command telling `Screen` what to do
`Nscreen`	which monitor to use for displays

Returned Variables:

`Wptr`	pointer/handle to the window opened
`rect`	dimensions of the window

Screen() command

`rect` dimensions of the window

```
>> rect
rect =
           0     0     1280     1024
```

Screen() command

```
Screen('Close', wPtr);   % close a particular window
```

`'Close'` command telling `Screen` what to do

`wPtr` pointer/handle to the window to be closed

```
Screen('CloseAll');      % close all open windows
```

getting back to Matlab

If Matlab freezes or otherwise gets stuck, and you're between 'OpenWindow' and 'Close' and you are operating on a single monitor, you will need to get back to Matlab.

Mac	PC
control-C	Ctrl-C
command-0	Alt-Tab to Matlab
`>> clear Screen`	`>> clear Screen`
`>> clear all`	`>> clear all`
`>> clear mex`	`>> clear mex`
option-command-Esc	Ctrl-Alt-Delete

Screen() command

```
try
   Nscreen = 0;
   [wPtr, rect] = Screen('OpenWindow', Nscreen);

   % simulate an error here
   i = -1;     a(i) = 99;

   Screen('Close', wPtr);

catch
   fclose('all'); Screen('CloseAll');
   ShowCursor; ListenChar;
   Priority(0);  psychrethrow(psychlasterror);
   disp('Error!');
end
```

Screen() command

```
Nscreen = 1;

% basic OpenWindow command
[wPtr, rect] = Screen('OpenWindow', Nscreen);

% what color is black
black = BlackIndex(wPtr);

% draw to fill the screen with black, offscreen
Screen('FillRect', wPtr, black, [100 100 500 500]);

% flip the offscreen to onscreen
Screen(wPtr, 'Flip');

% pause
pause(1);

% close the monitor window
Screen('Close', wPtr);
```

what's "Black" or "White" could depend on the graphics setting, so doing this is best.

Screen() command

```
Nscreen = 1;

% basic OpenWindow command
[wPtr, rect] = Screen('OpenWindow', Nscreen);

% what color is black
black = BlackIndex(wPtr);

% draw to fill the screen with black, offscreen
Screen('FillRect', wPtr, black, [100 100 500 500]);

% flip the offscreen to onscreen
Screen(wPtr, 'Flip');

% pause
pause(1);

% close the monitor window
Screen('Close', wPtr);
```

Screen() command

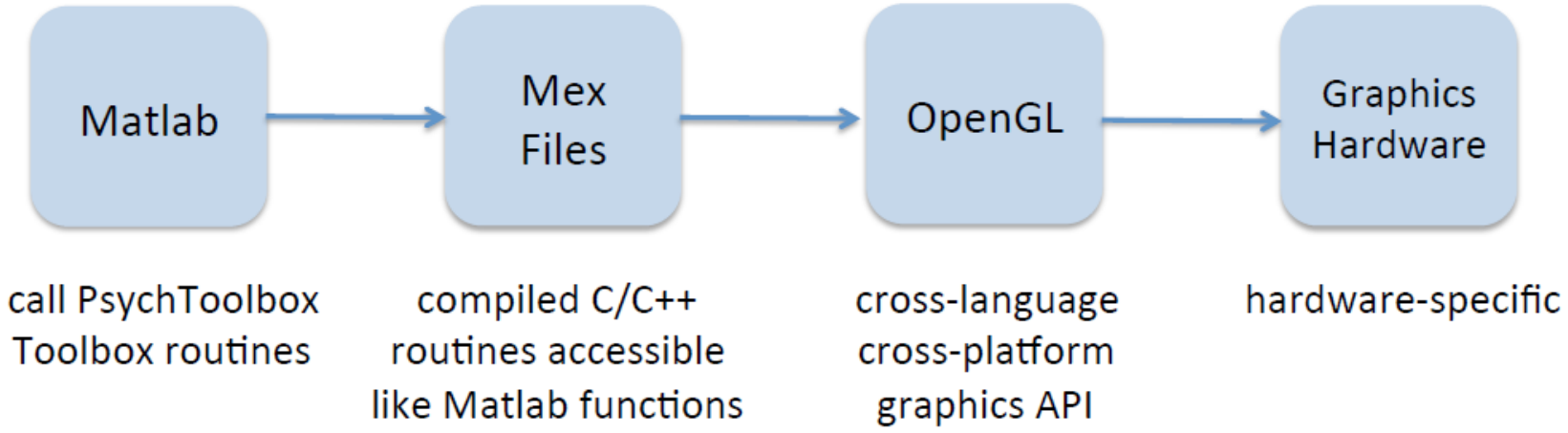
```
Nscreen = 1;

% basic OpenWindow command
[wPtr, rect] = Screen('OpenWindow', Nscreen);

% what color is black
black = BlackIndex(wPtr);

% draw to fill the screen with black, offscreen
Screen('FillRect', wPtr, black, [100 100 500 500]);

% flip the offscreen to onscreen
Screen(wPtr, 'Flip');

% pause
pause(1);

% close the monitor window
Screen('Close', wPtr);
```

Screen() command

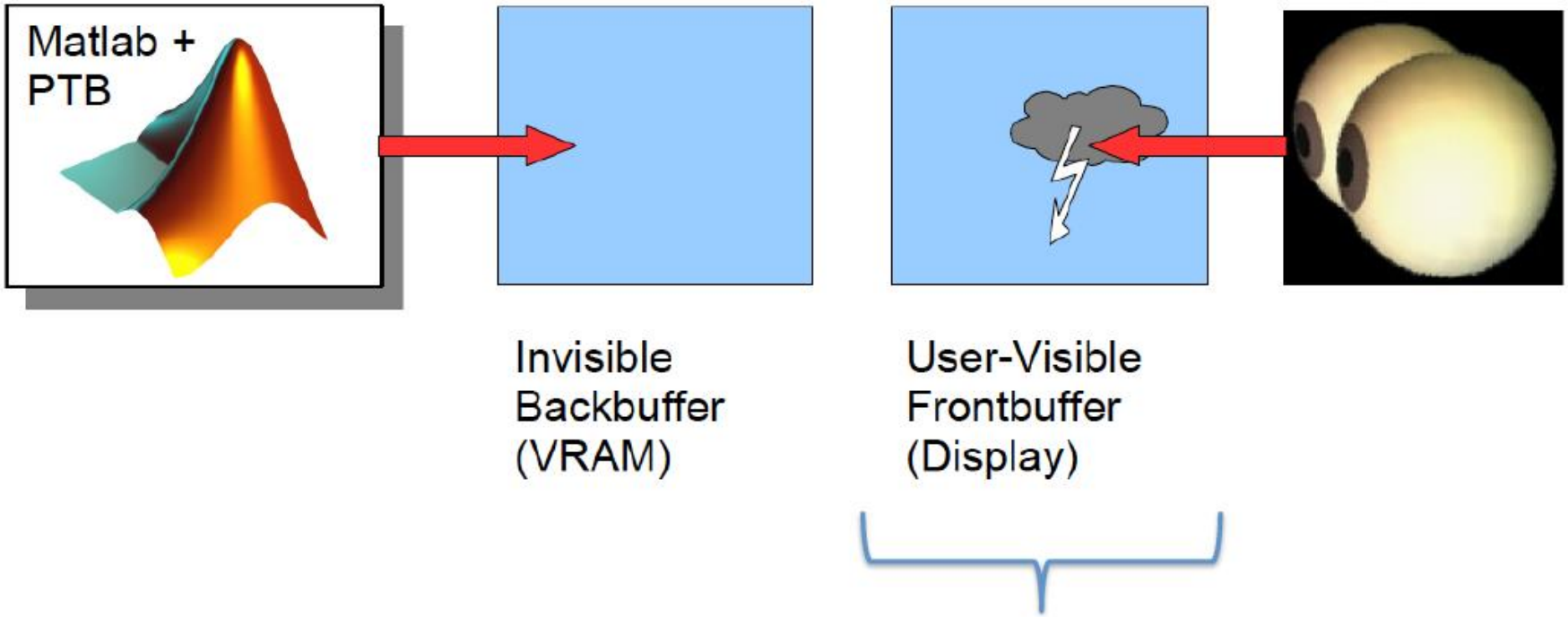
```
Nscreen = 0;
[wPtr, rect] = Screen('OpenWindow', Nscreen);

white = WhiteIndex(wPtr);
Screen('FillRect', wPtr, white);

% draw a rectangle in the middle, offscreen
cx = rect(1) + (rect(3)-rect(1))/2;
cy = rect(2) + (rect(4)-rect(2))/2;
s = 250;
Screen('FillRect', wPtr, black, [cx-s cy-s cx+s cy+s]);

% flip
Screen(wPtr, 'Flip');
pause(1);
Screen('Close', wPtr);
```

try running without 'Flip' to see what is DOESN'T do

Behind the scenes

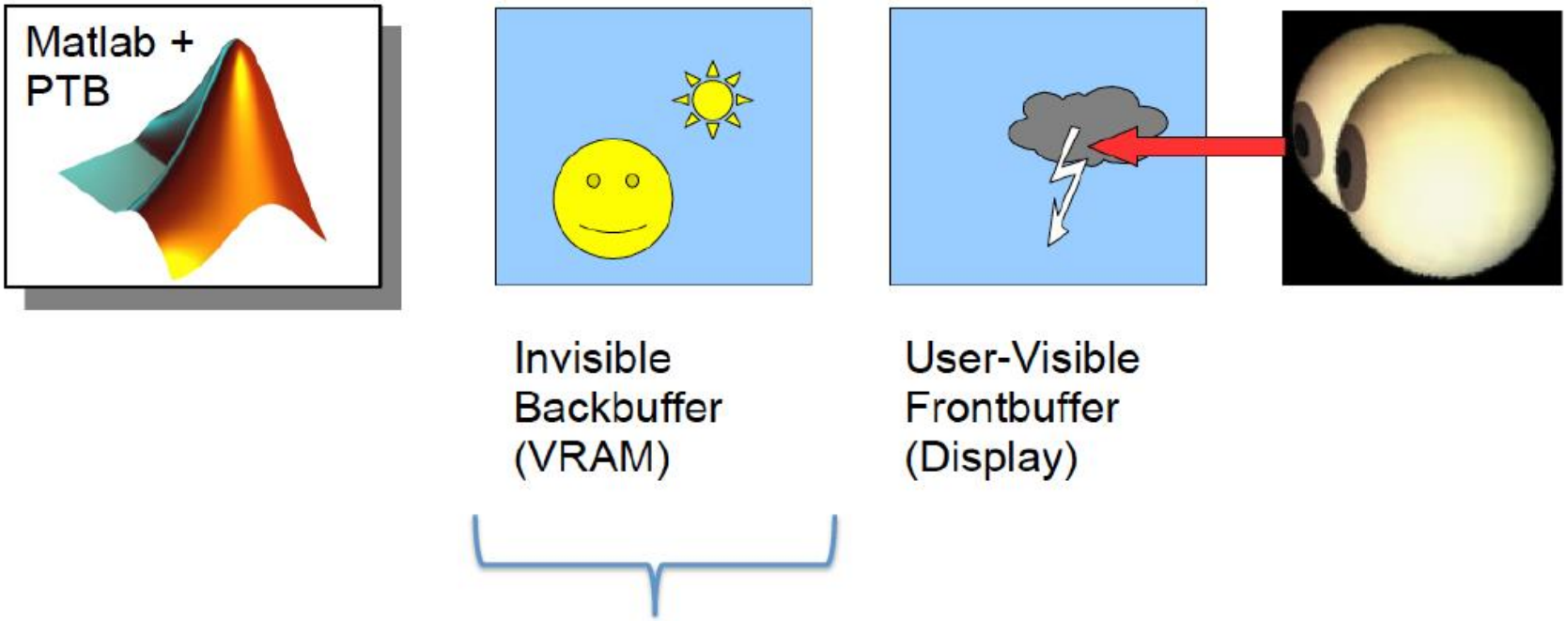


Behind the scenes

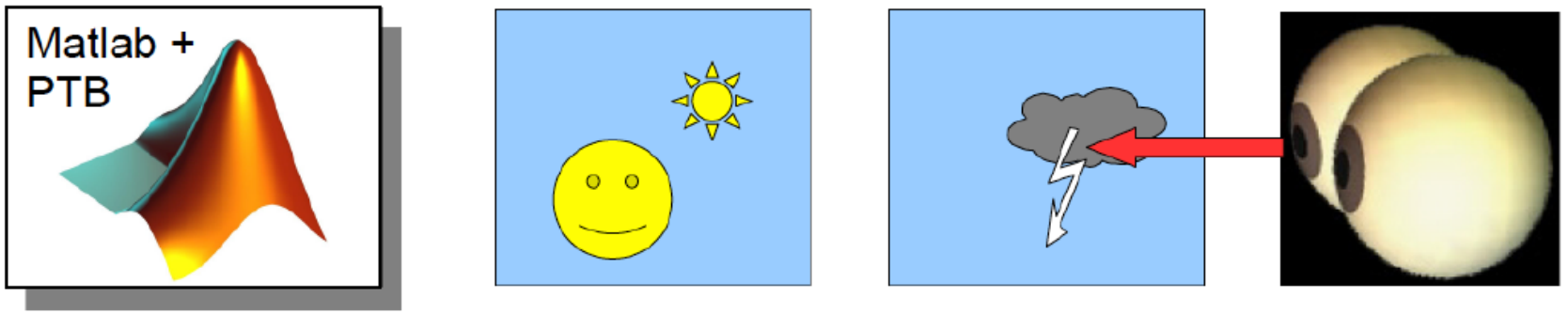


some slide adapted from:
What's new in Psychtoolbox-3?
A free cross-platform toolkit for Psychophysics
with Matlab & GNU/Octave
Mario Kleiner, David Brainard, Denis Pelli

Behind the scenes

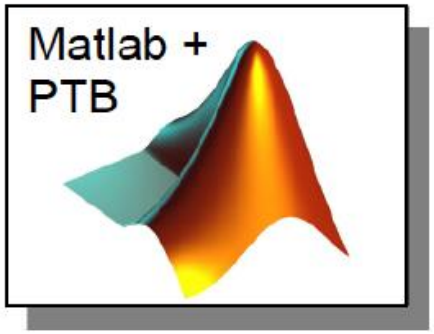


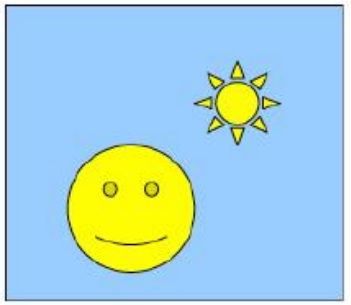
Behind the scenes



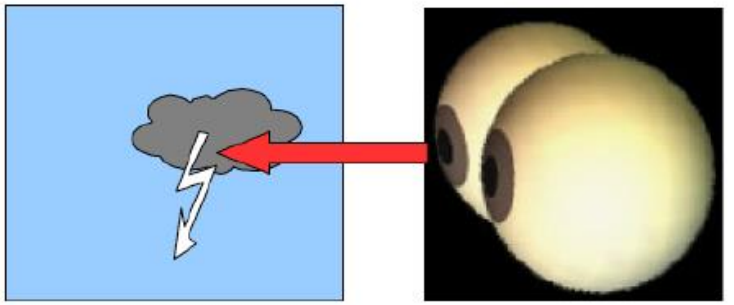
writing from computer memory (RAM) to graphics memory (VRAM) takes time – OpenGL performs that copying in the background while Matlab and PTB let you do other things

Behind the scenes





Invisible
Backbuffer
(VRAM)

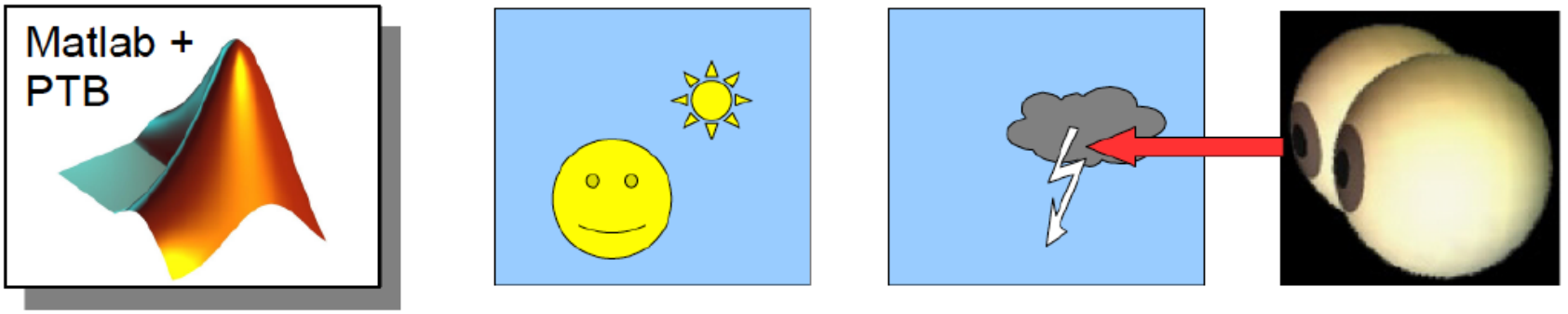


User-Visible
Frontbuffer
(Display)

issue `Screen('Flip')` command to flip the invisible backbuffer to the visible frontbuffer

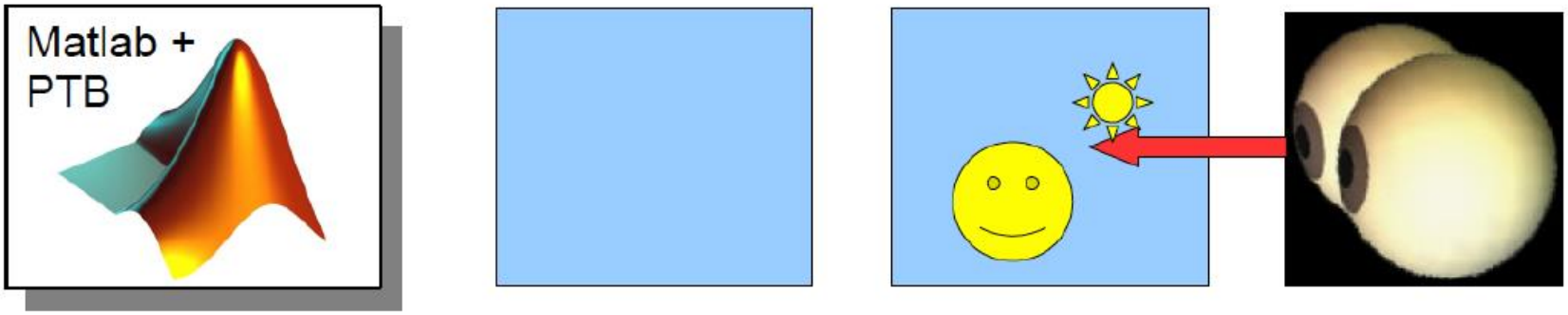
graphics hardware waits to flip to avoid artifacts

Behind the scenes



1) the backbuffer must be filled completely
2) the monitor must not be in the midst of a retrace

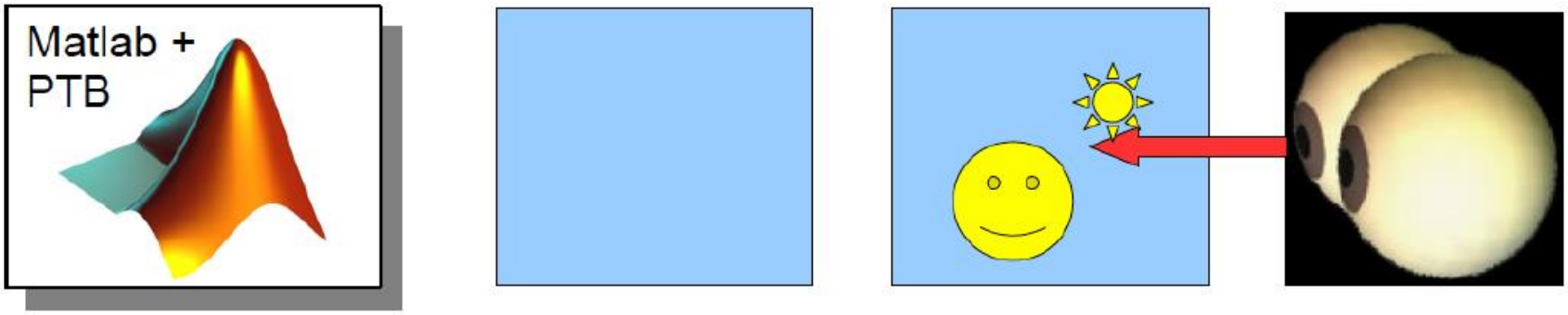
Behind the scenes



subject perceives a "tear-free" monitor update

and you get sub-ms accurate timestamp of when the monitor change occurred

Behind the scenes



backbuffer is cleared to background for next update

(you can also Flip and keep the backbuffer as is to add to it)