دانشکده روانشناسی و علوم تربیتی

IPM
پژوهشگاه دانش‌های بنیادی

MATLAB for Brain and Cognitive Psychology (Presentation stimuli)

Presented by:
Ehsan Rezayat, Ph.D.
Faculty of Psychology and Education, University of Tehran.
Institute for Research in Fundamental Sciences (IPM), School of Cognitive Sciences,
emails: rezayat@ut.ac.ir, rezayat@ipm.ir, erezayat.er@gmail.com

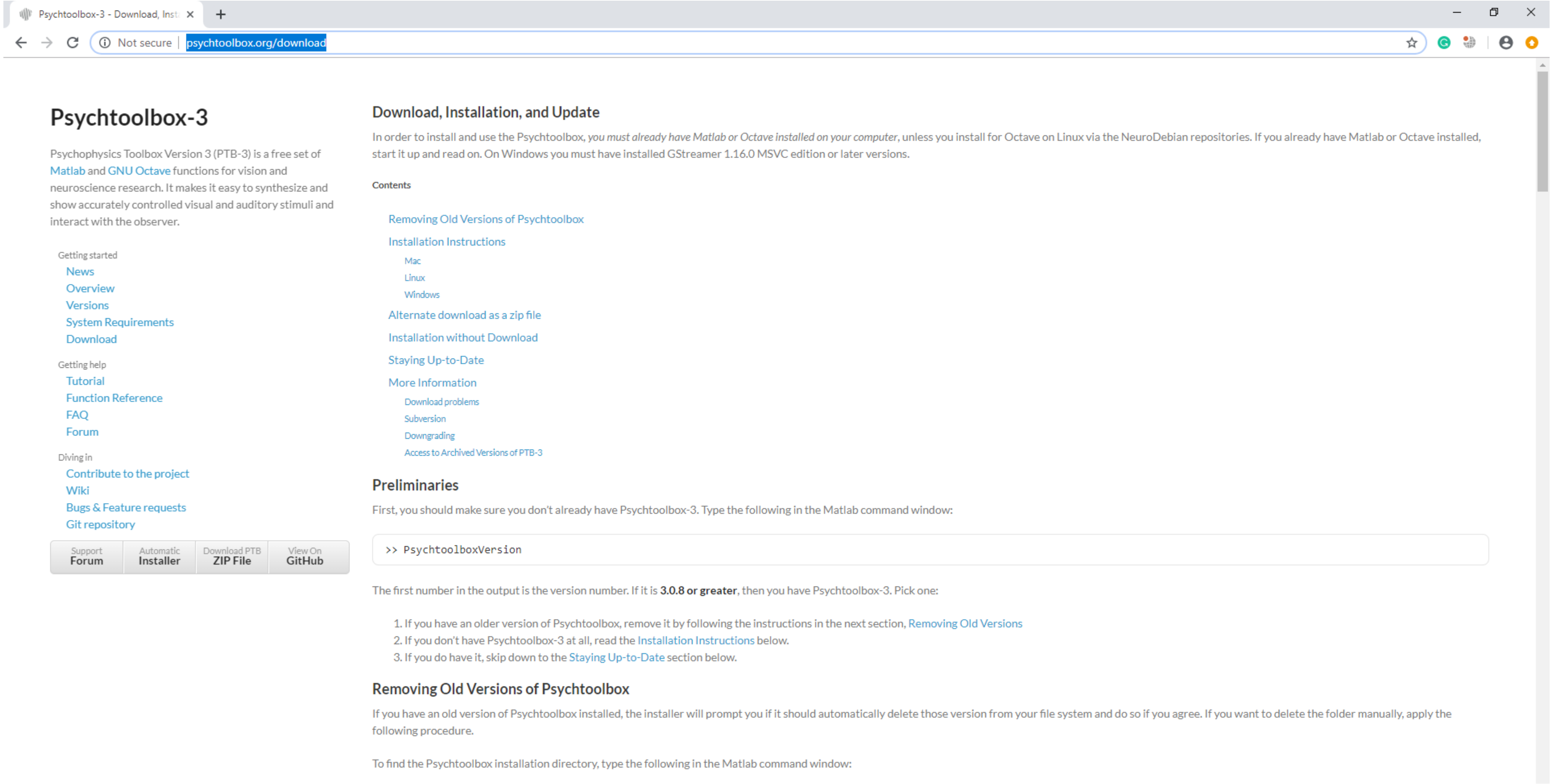
Today: Steps in the Psychophysics Lab

- Generating Stimuli (today)
- Stimulus presentation
- Visual Display
- Response collection

PsychToolbox (PTB)

- A set of Matlab functions written by some vision researchers.
- Not written by Mathworks.
- Mature code: started in 1995, current version is PTB-3
- Brainard, D. H. (1997) The Psychophysics Toolbox, *Spatial Vision 10*:433-436

- http://psychtoolbox.org/download



Psychtoolbox-3

Psychophysics Toolbox Version 3 (PTB-3) is a free set of Matlab and GNU Octave functions for vision and neuroscience research. It makes it easy to synthesize and show accurately controlled visual and auditory stimuli and interact with the observer.

Getting started
- News
- Overview
- Versions
- System Requirements
- Download

Getting help
- Tutorial
- Function Reference
- FAQ
- Forum

Diving in
- Contribute to the project
- Wiki
- Bugs & Feature requests
- Git repository

Download, Installation, and Update

In order to install and use the Psychtoolbox, *you must already have Matlab or Octave installed on your computer*, unless you install for Octave on Linux via the NeuroDebian repositories. If you already have Matlab or Octave installed, start it up and read on. On Windows you must have installed GStreamer 1.16.0 MSVC edition or later versions.

Contents

Preliminaries

First, you should make sure you don't already have Psychtoolbox-3. Type the following in the Matlab command window:

```
>> PsychtoolboxVersion
```

The first number in the output is the version number. If it is **3.0.8 or greater**, then you have Psychtoolbox-3. Pick one:

1. If you have an older version of Psychtoolbox, remove it by following the instructions in the next section, Removing Old Versions
2. If you don't have Psychtoolbox-3 at all, read the Installation Instructions below.
3. If you do have it, skip down to the Staying Up-to-Date section below.

Removing Old Versions of Psychtoolbox

If you have an old version of Psychtoolbox installed, the installer will prompt you if it should automatically delete those version from your file system and do so if you agree. If you want to delete the folder manually, apply the following procedure.

To find the Psychtoolbox installation directory, type the following in the Matlab command window:

Testing your PTB installation

```
>> PsychtoolboxVersion
ans =

3.0.12 - Flavor: beta - Corresponds to SVN Revision 7762
For more info visit:
https://github.com/Psychtoolbox-3/Psychtoolbox-3

>> UpdatePsychtoolbox
>>
>> help PsychDemos
>>
>> KbDemo
```

Before you start

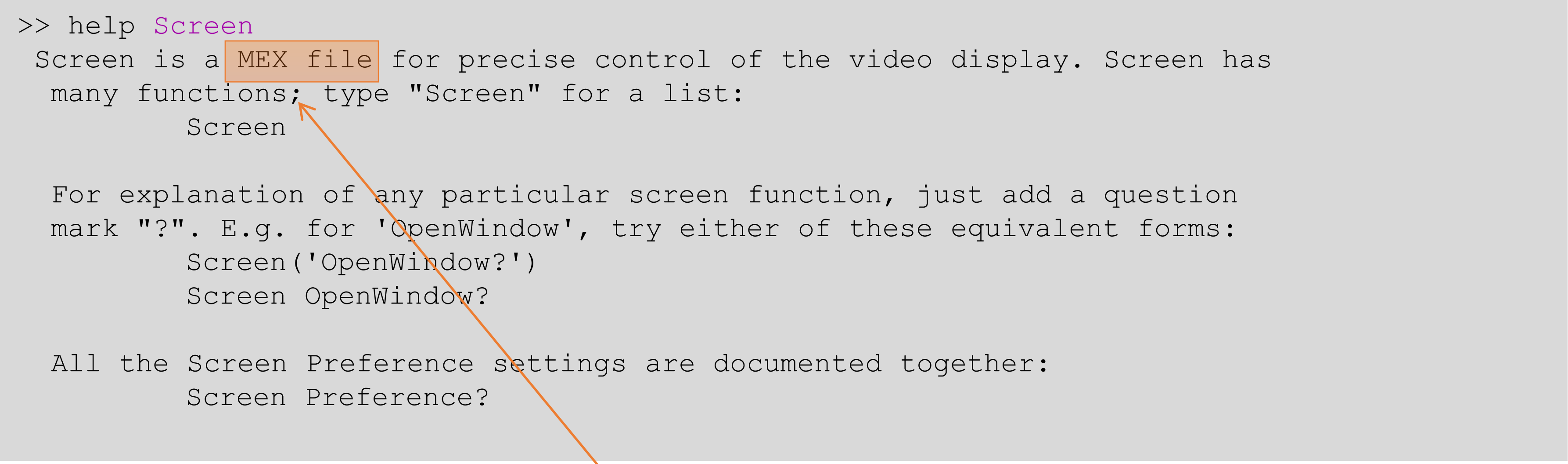
```
>> ScreenTest
PTB-INFO: Connection to Psychtoolbox kernel support driver instance #0 (Revision 0) established.
PTB-INFO: Connection to Psychtoolbox kernel support driver instance #1 (Revision 0) established.
PTB-INFO: Switching to kernel driver instance #1 in hybrid graphics system, assuming i am attached to discrete
non-Intel GPU.
***** ScreenTest: Testing Screen 0 *****


PTB-INFO: This is Psychtoolbox-3 for Apple OS X, under Matlab 64-Bit (Version 3.0.11 - Build date: Jul  8 2013).
PTB-INFO: Type 'PsychtoolboxVersion' for more detailed version information.
PTB-INFO: Most parts of the Psychtoolbox distribution are licensed to you under terms of the MIT License, with
PTB-INFO: some restrictions. See file 'License.txt' in the Psychtoolbox root folder for the exact licensing
conditions.

PTB-INFO: Deficient Apple OS/X 10.7 or later detected: Would use fragile CoreVideo timestamping as fallback,
PTB-INFO: if beamposition timestamping would not work. Will try to use beamposition timestamping if possible.


PTB-INFO: OpenGL-Renderer is NVIDIA Corporation :: NVIDIA GeForce GT 330M OpenGL Engine :: 2.1 NVIDIA-8.12.47
310.40.00.05f01
PTB-INFO: Renderer has 512 MB of VRAM and a maximum 487 MB of texture memory.
PTB-INFO: VBL startline = 1050 , VBL Endline = 1079
PTB-INFO: Measured monitor refresh interval from beamposition = 16.699159 ms [59.883255 Hz].
PTB-INFO: Will use beamposition query for accurate Flip time stamping.
PTB-INFO: Measured monitor refresh interval from VBLsync = 16.672991 ms [59.977240 Hz]. (54 valid samples taken,
stddev=0.198526 ms.)
PTB-INFO: Small deviations between reported values are normal and no reason to worry.
PTB-INFO: Support for fast OffscreenWindows enabled.

***** ScreenTest: Done With Screen 0 *****
```

Installing the kernel driver

- On OSX there is a kernel extension (.kext) that you can install to make screen timing more precise
- type help PsychtoolboxKernelDriver for instructions on how to install (basically you just unzip the kernel driver into the right system folder)

The Screen command

- Screen() is the heart of Psychtoolbox



```
>> help Screen
 Screen is a MEX file for precise control of the video display. Screen has
  many functions; type "Screen" for a list:
          Screen

  For explanation of any particular screen function, just add a question
  mark "?". E.g. for 'OpenWindow', try either of these equivalent forms:
          Screen('OpenWindow?')
          Screen OpenWindow?

  All the Screen Preference settings are documented together:
          Screen Preference?
```

MEX = "Matlab Executable"
A file written in another language like C
that can be called as a Matlab function

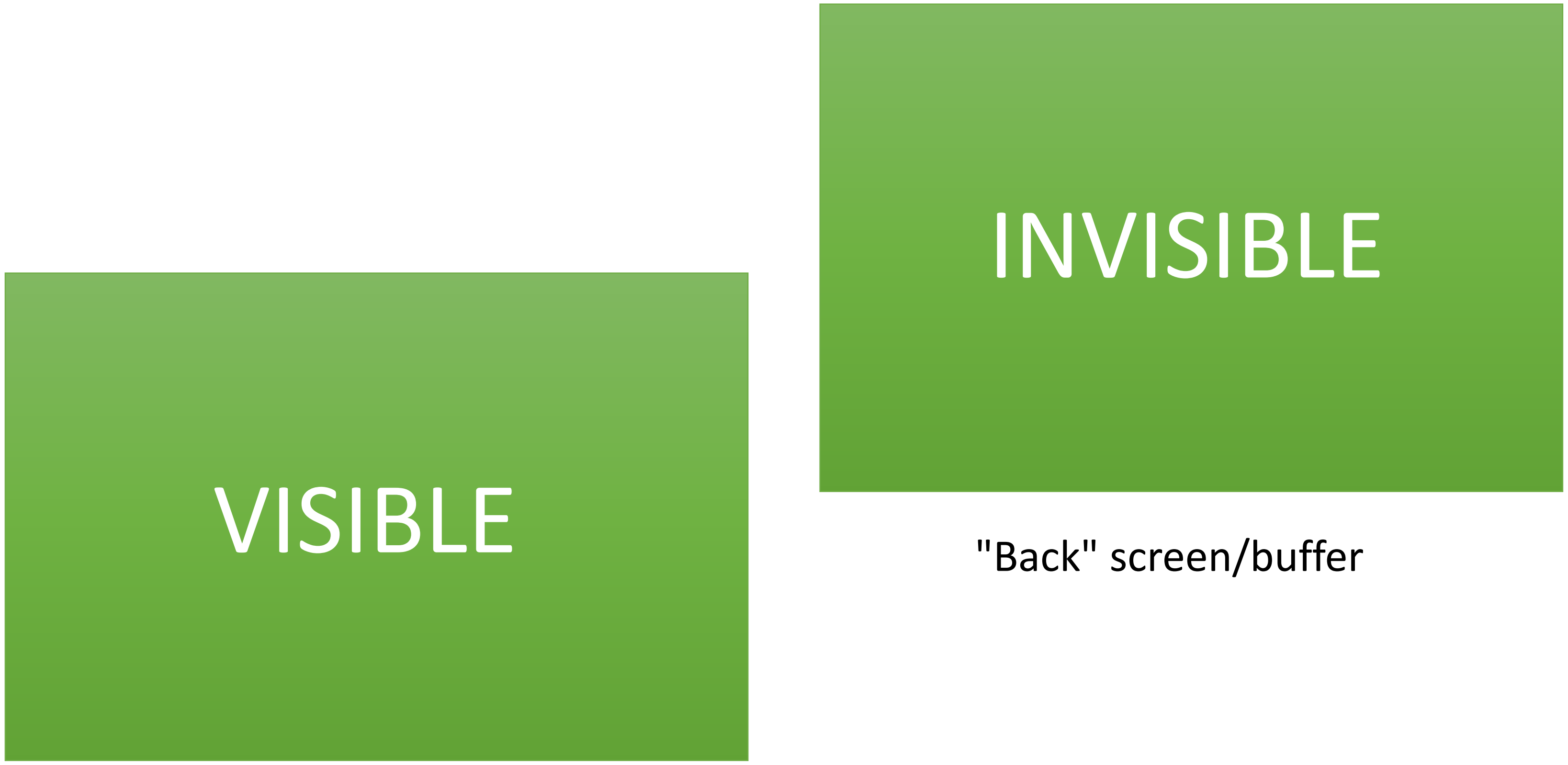
The Screen command

```
>> Screen
Usage:

% Activate compatibility mode: Try to behave like the old MacOS-9 Psychtoolbox:
oldEnableFlag=Screen('Preference', 'EmulateOldPTB', [enableFlag]);

% Open or close a window or texture:
[windowPtr,rect]=Screen('OpenWindow',windowPtrOrScreenNumber [,color] [,rect] [,pixelSize] [,numberOfBuffers]
[,stereomode] [,multisample][,imagingmode][,specialFlags][,clientRect]);
[windowPtr,rect]=Screen('OpenOffscreenWindow',windowPtrOrScreenNumber [,color] [,rect] [,pixelSize]
[,specialFlags] [,multiSample]);
textureIndex=Screen('MakeTexture', WindowIndex, imageMatrix [, optimizeForDrawAngle=0] [, specialFlags=0] [,
floatprecision=0] [, textureOrientation=0] [, textureShader=0]);
oldParams = Screen('PanelFitter', windowPtr [, newParams]);
Screen('Close', [windowOrTextureIndex or list of textureIndices/offscreenWindowIndices]);
Screen('CloseAll');

%  Draw lines and solids like QuickDraw and DirectX (OS 9 and Windows):
currentbuffer = Screen('SelectStereoDrawBuffer', windowPtr [, bufferid] [, param1]);
Screen('DrawLine', windowPtr [,color], fromH, fromV, toH, toV [,penWidth]);
Screen('DrawArc',windowPtr,[color],[rect],startAngle,arcAngle)
Screen('FrameArc',windowPtr,[color],[rect],startAngle,arcAngle[,penWidth] [,penHeight] [,penMode])
Screen('FillArc',windowPtr,[color],[rect],startAngle,arcAngle)
Screen('FillRect', windowPtr [,color] [,rect] );
Screen('FrameRect', windowPtr [,color] [,rect] [,penWidth]);
Screen('FillOval', windowPtr [,color] [,rect] [,perfectUpToMaxDiameter]);
Screen('FrameOval', windowPtr [,color] [,rect] [,penWidth] [,penHeight] [,penMode]);
Screen('FramePoly', windowPtr [,color], pointList [,penWidth]);
```

The Screen command

```
>> Screen DrawLine?
Usage:

Screen('DrawLine', windowPtr [,color], fromH, fromV, toH, toV [,penWidth]);

Draw a line. "color" is the clut index (scalar or [r g b a] vector) that you
want to poke into each pixel; default produces black. "fromH" and "fromV" are
the starting x and y positions, respectively. "toH" and "toV" are the ending x
and y positions, respectively. Default "penWidth" is 1.

See also: DrawLines
```

Double buffering



INVISIBLE

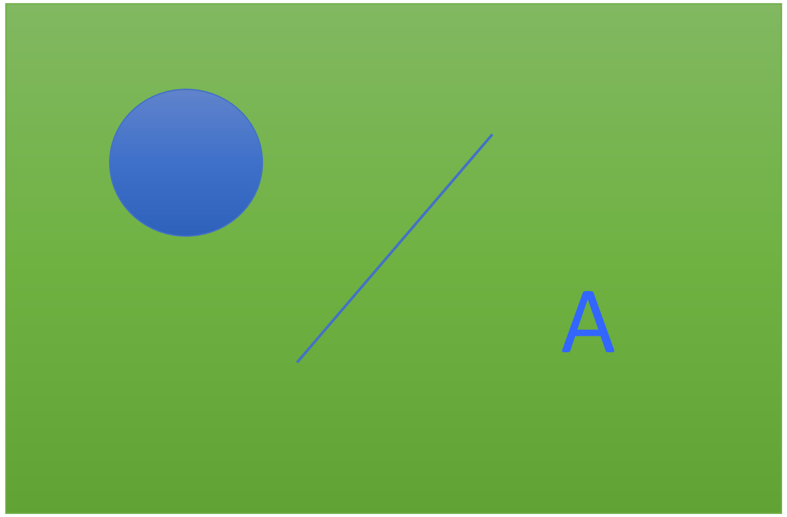
"Back" screen/buffer

VISIBLE

"Front" screen/buffer

Double buffering

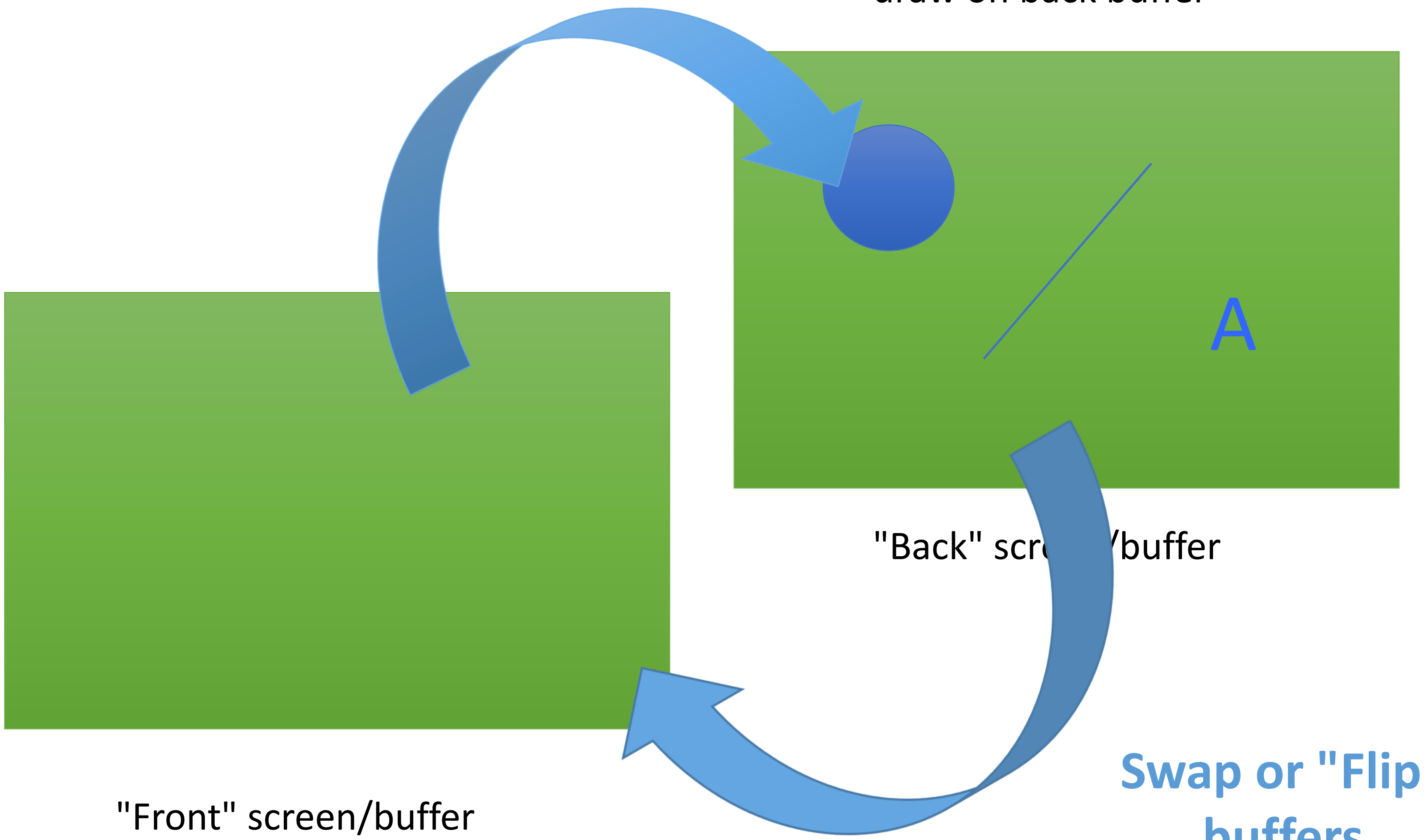
draw on back buffer



"Back" screen/buffer

"Front" screen/buffer

Double buffering



draw on back buffer

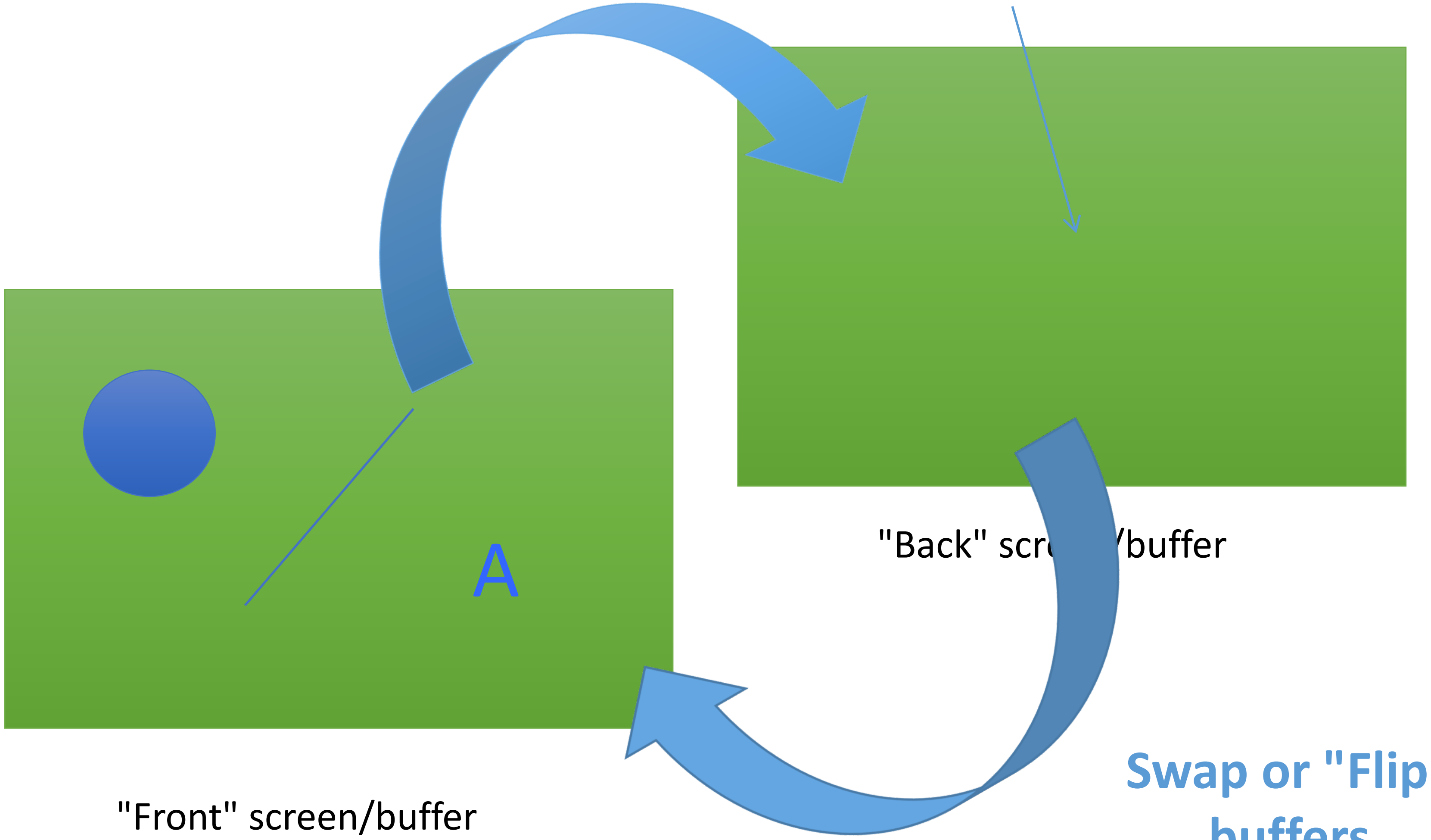
"Back" screen/buffer

"Front" screen/buffer

Swap or "Flip" buffers

Double buffering

Note that flipping clears
the new back buffer



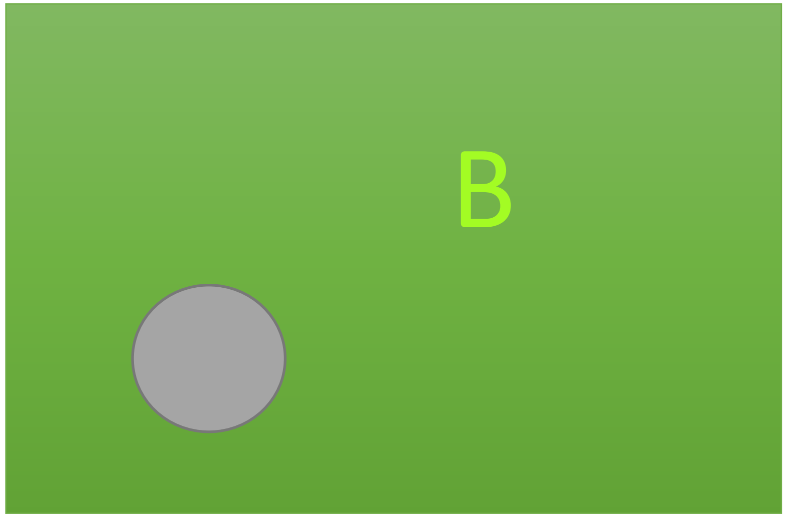
"Back" screen/buffer

"Front" screen/buffer

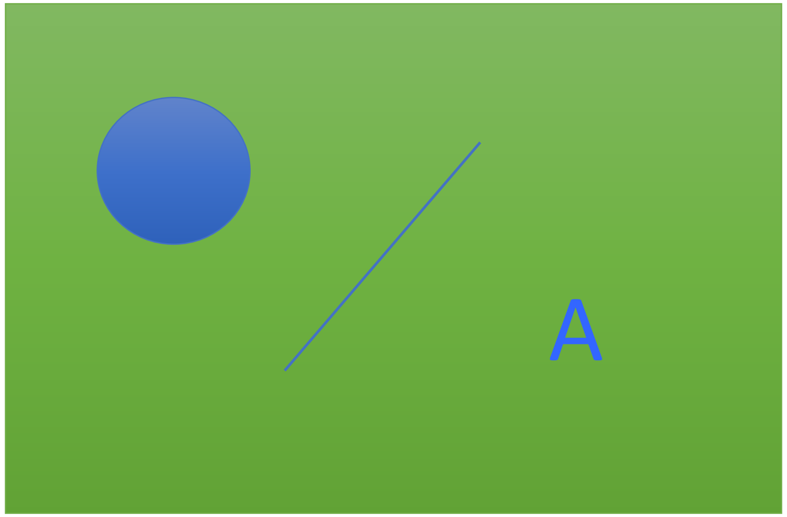
**Swap or "Flip"
buffers**

Double buffering

Draw next frame on back buffer

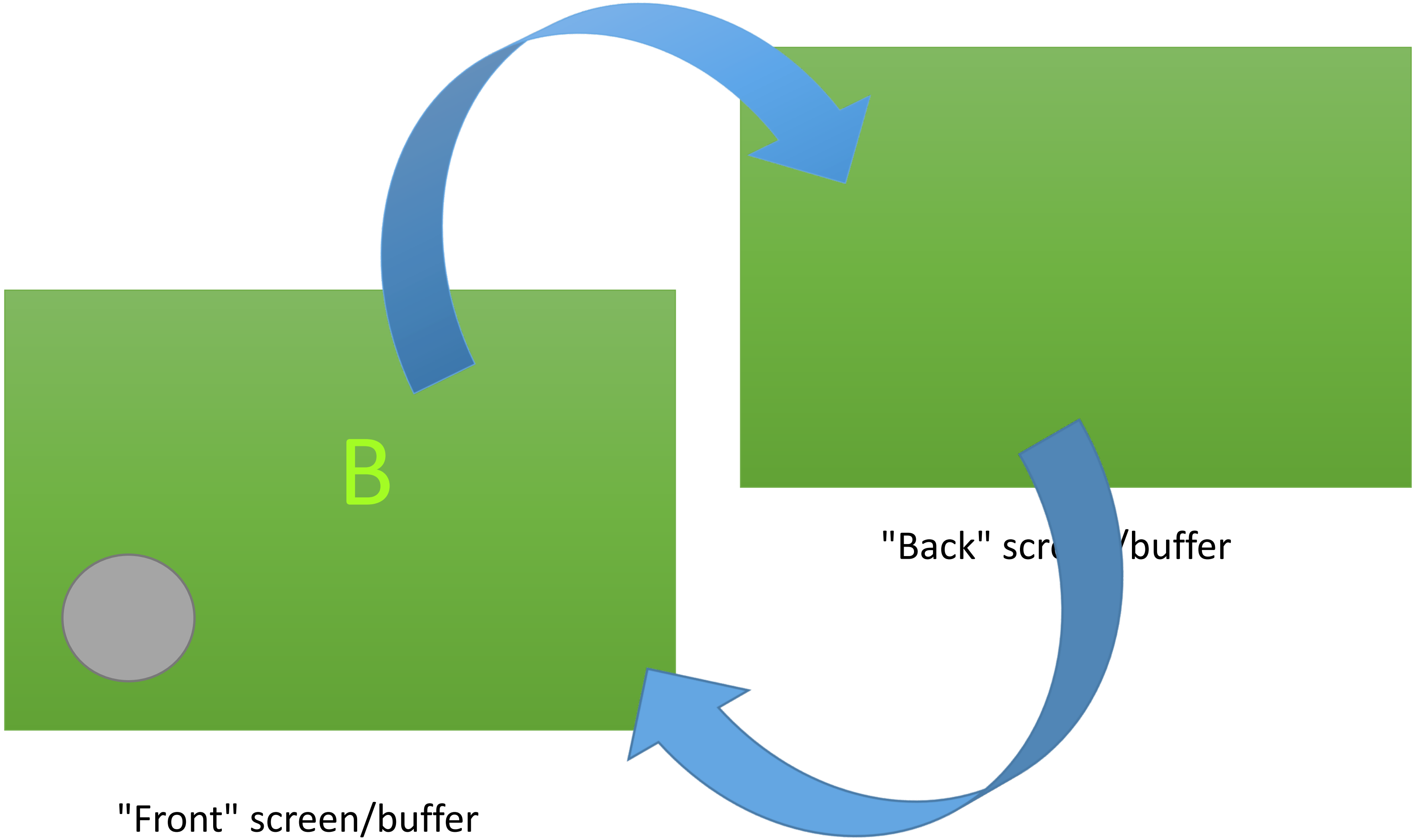


"Back" screen/buffer



"Front" screen/buffer

Double buffering



Flip again

B

"Back" scr /buffer

"Front" screen/buffer

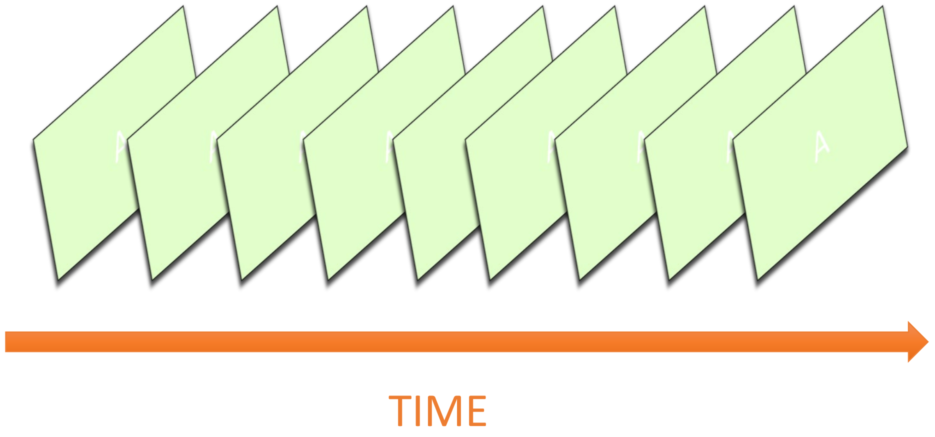
How monitors work

CRT
(Cathode Ray Tube)

LCD
(Liquid Crystal Display)

How monitors work

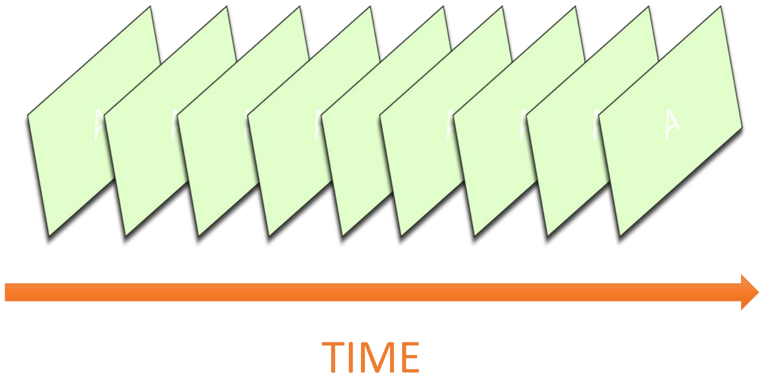
Frame Rate: The number of frames drawn per second



TIME

Typical frame rate: 60Hz (60 frames per second)
1 second / 60 frames == **16.67 milliseconds per frame**

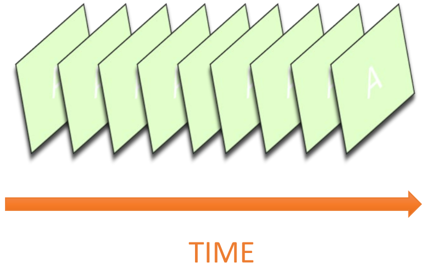
How monitors work



Frame Rate

- Puts limits on the precision of our visual presentation
- Cannot present something for shorter than the length of a single frame
- Screen refresh timing is the anchor that PTB uses for all timing measurement

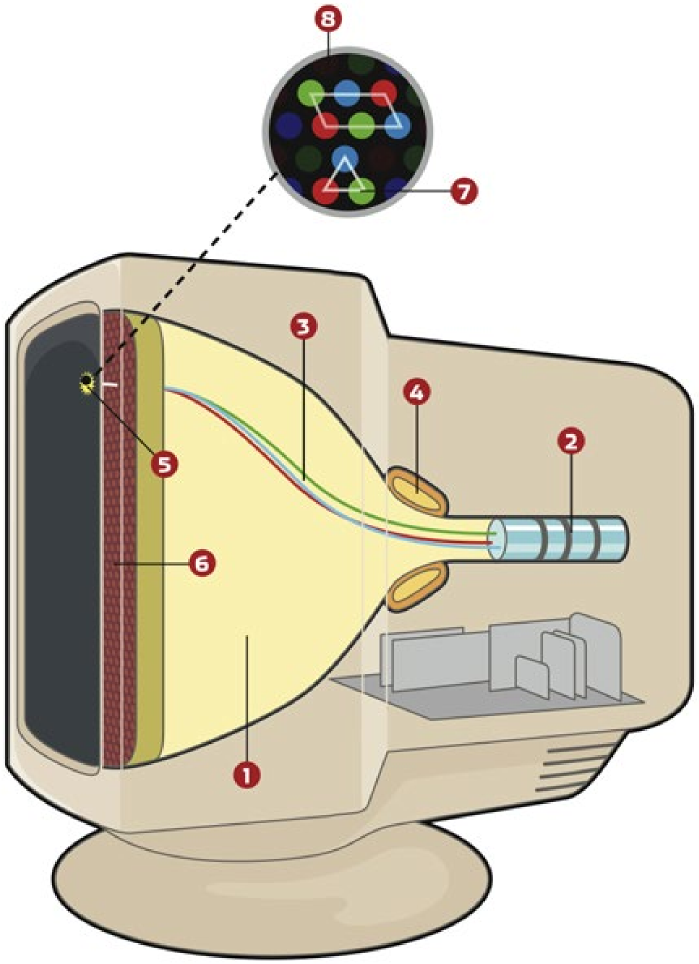
How monitors work



Getting your frame duration in PTB

```
frameDuration = Screen('GetFlipInterval',windowPtr)
```

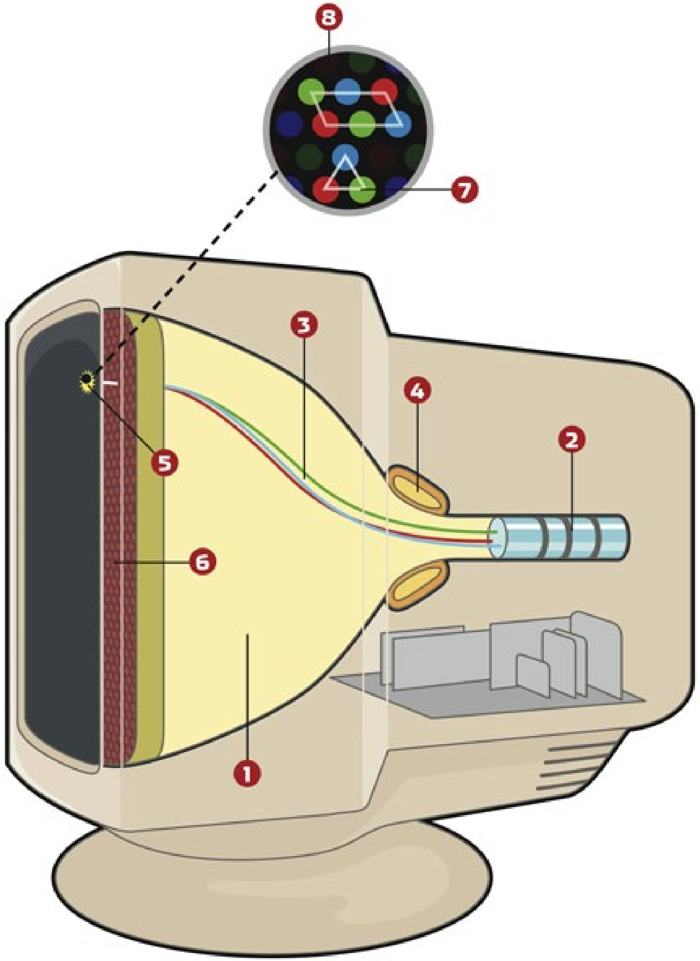
How monitors work



1 – cathode ray tube
2 – electron gun
3 – electron beam
4 – deflection yoke

The deflection yoke manipulates the electron beam, sweeping it across the screen, one horizontal line ("scanline") at a time

How monitors work



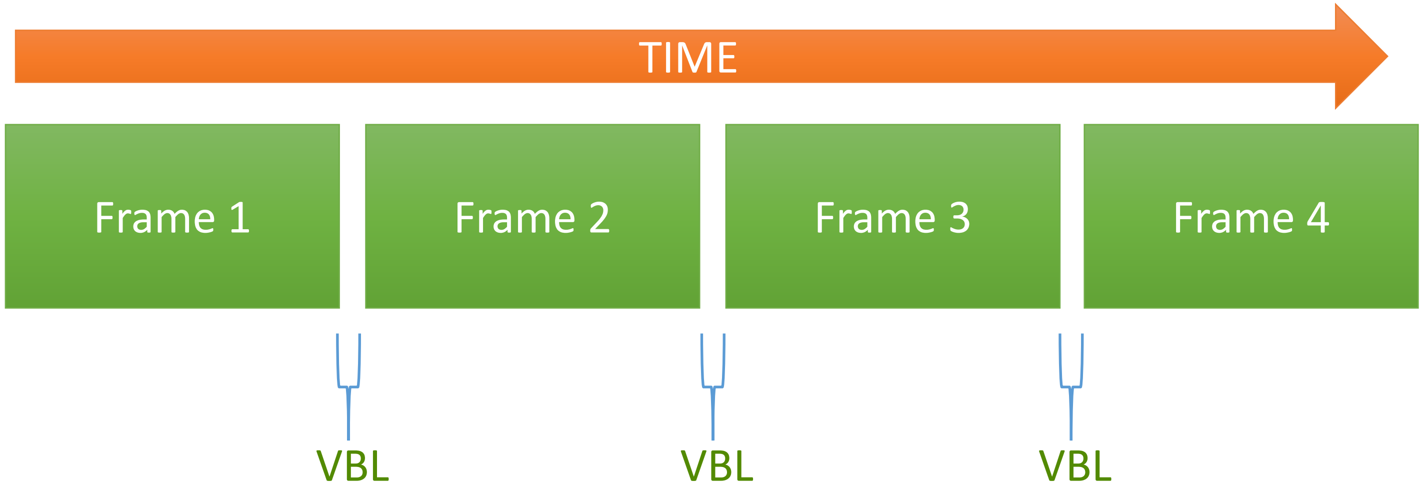
Once one frame is completely drawn, there is a gap in time as the beam is blanked and sweeps back to the first scanline to start drawing the next frame. This gap between frames is called the **Vertical BLank interval (VBL)**.

The current position of the beam while it scans is called **beamposition**.

How monitors work

To maintain backwards compatibility, LCD's also implement a **VBL** even though they don't technically need one.

They also report a **beamposition** (the location of the current scanline), even though they don't use a beam.

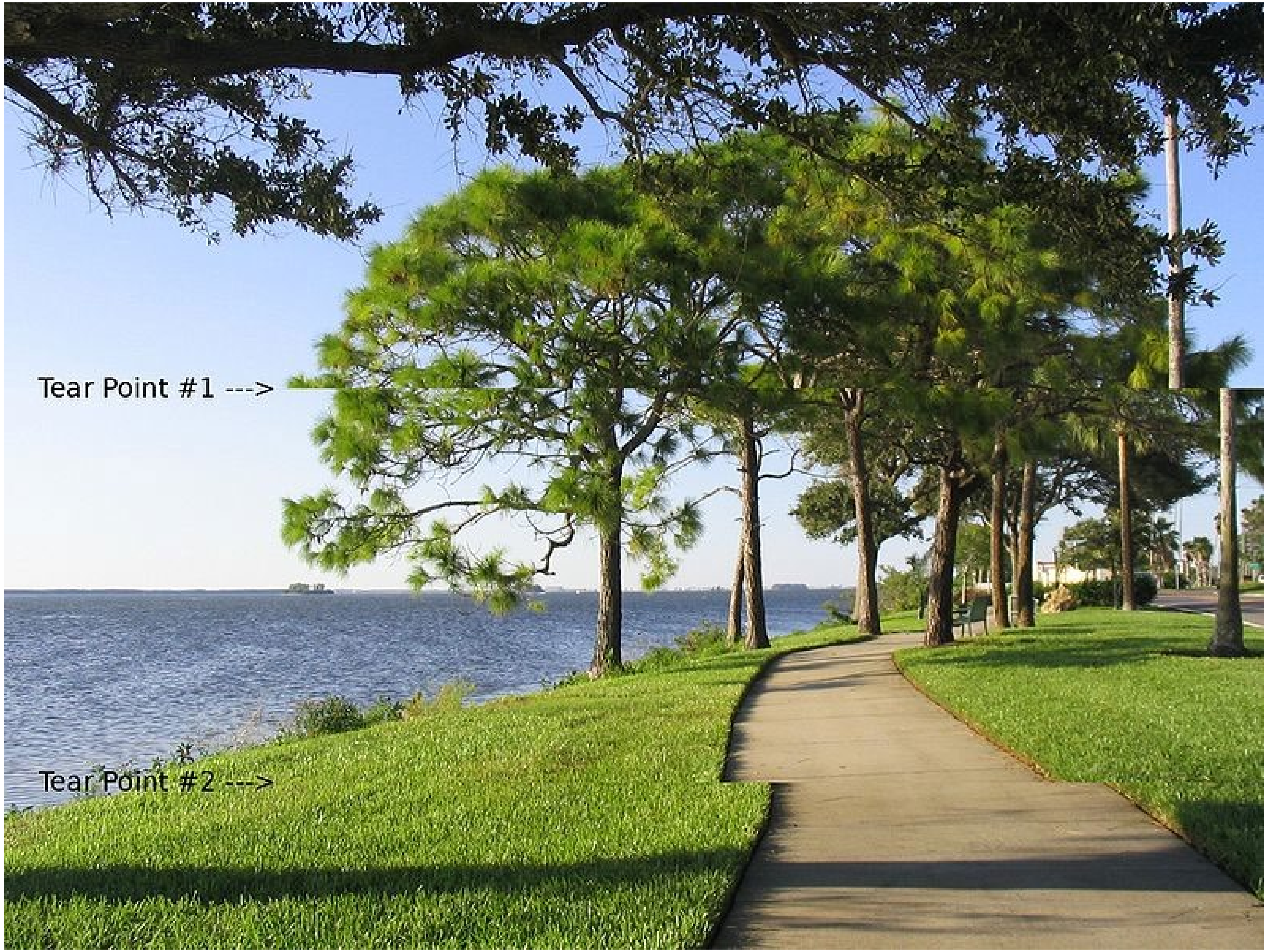


PTB tries to swap the front and back buffers during the VBL, so that content is not being updated in the middle of a frame draw.

This is called VBL Synchronization. If synchronization between buffer-swapping and VBL fails:

- Visual artifacts like flicker and tearing may occur
- Timing measurement will be less precise

Tearing artifact



Using the Screen command

- Whenever you draw to the screen in PTB, you are drawing to the back buffer
- You will not see anything you've drawn until you "Flip" the buffers
- This separates drawing and arranging time from presentation time – you can wait until the precise moment you want everything to appear and pull the trigger (Flip)

Using the Screen command

- Opening the screen

```
[windowPtr,rect]=Screen('OpenWindow',ScreenNumber)
```

returns a number that we will use to refer to this screen in future commands

which screen you want to open (you may have multiple monitors)

returns a rectangle (a vector of four numbers) that describe the dimensions of the screen

```
[windowPtr,rect]=Screen('OpenWindow',ScreenNumber)
```

which screen you want to open (you may have multiple monitors)

Slight OS difference here!

For all platform differences, see http://psychtoolbox.org/PlatformDifferences

```
>> Screen('Screens')

ans =

     0     1
```

MAC OS X:

0 is the main display (with the menubar) and 1 is the first external display

WINDOWS:

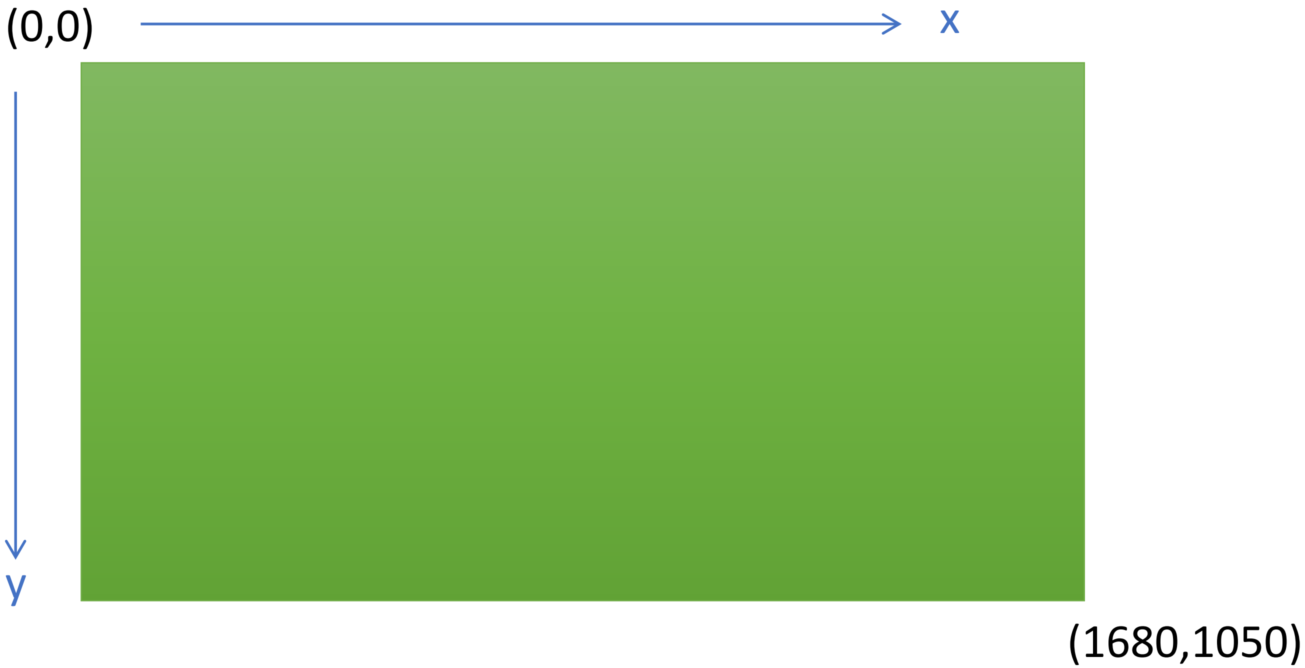
0 refers to all displays together, then 1 is the main monitor and 2-x are externals

```
>> max(Screen('Screens'))

ans =

     1
```

```
[windowPtr,rect]=Screen('OpenWindow',ScreenNumber)
```

returns a rectangle (a vector of four numbers)
that describe the dimensions of the screen

```
>> rect

rect =                          width      height

          0          0          1680       1050
```



Using Screen

```
function drawSomething()

[wPtr, rect] = Screen('OpenWindow',max(Screen('Screens')));  %open the screen
Screen('FillRect', wPtr, [255 0 0],[100 100 500 500]);       %draw a rectangle on the back buffer
Screen('Flip',wPtr);                                         %flip the buffers
KbWait();                                                    %wait until key pressed

clear Screen;

end
```

Waiting

```
WaitSecs(s)

WaitSecs('UntilTime',when)

KbWait()
```

KbWait

```
[secs, keyCode, deltaSecs] = KbWait()
```

Will wait until the user presses a key, and return the time and keypress.

KbWait IS NOT FOR MEASURING REACTION TIMES!!
(It polls every 5 ms)

KbWait

[secs, keyCode, deltaSecs] = KbWait()

keyCode is a vector of all the keys, with a 1 for any key that was pressed.

find(keyCode) will return the index of the button(s) pressed.

That code can then be turned into a character using KbName()

KbWait

```
>> WaitSecs(1);[secs, keyCode, deltaT] = KbWait();
>> find(keyCode)

ans =

    32

>> KbName(32)

ans =

3#
>> KbName('3#')

ans =

    32
```

Drawing in PTB

PTB uses OpenGL for drawing to the screen.

Open GL is the "Open Graphics Library". It is cross-platform software for specifying how to draw 2D and 3D graphics using the GPU to achieve hardware-accelerated processing.

PTB has its own functions for drawing that access lower-level OpenGL functions. But if you want to access actual OpenGL commands you can do that too, or use OpenGL objects created in a program like Blender.

Testing the screen

- When you run Screen('OpenWindow'), PTB will go through a series of Sync Tests and will report to you any issues. Read this information carefully and follow its advice.
  - The flashing triangle warning generally means Sync has failed

- Several additional tools are available to test and diagnose screen sync issues:
  - ScreenTest()
  - VBLSyncTest()
  - PerceptualVBLSyncTest()

Testing the screen

- If timing is important to you, and you are having VBL sync issues, try the following:
  - If you are using multiple monitors, match their resolutions and settings, or use mirror mode
  - Only use one monitor
  - On mac, make sure PsychtoolboxKernelDriver is installed
  - read help SyncTrouble for other tips and Platform-specific issues

Skipping Sync Tests

- If Sync is not important to you, for instance you are debugging on a machine that you will not use for actual testing, you can disable the Sync test that is performed when you invoke OpenWindow:

```
Screen('Preference','SkipSyncTests',1);
```

↗ (you can set this value back to 0 to re-enable SyncTests)

Screen Timing

flipTime = Screen('Flip',windowPtr)

```
>> wPtr = Screen('OpenWindow',1);
>> flipTime = Screen('Flip',wPtr);
flipTime =

   1.1038e+05

>>
```

Screen Timing

GetSecs() tells you the current time

```
>> now = GetSecs()

now =

   1.1058e+05

>> aLittleLater = GetSecs()

aLitterLater =

         1.1060e+05

>> gap = aLittleLater - now

gap =

         21.2212
```

Flip timing

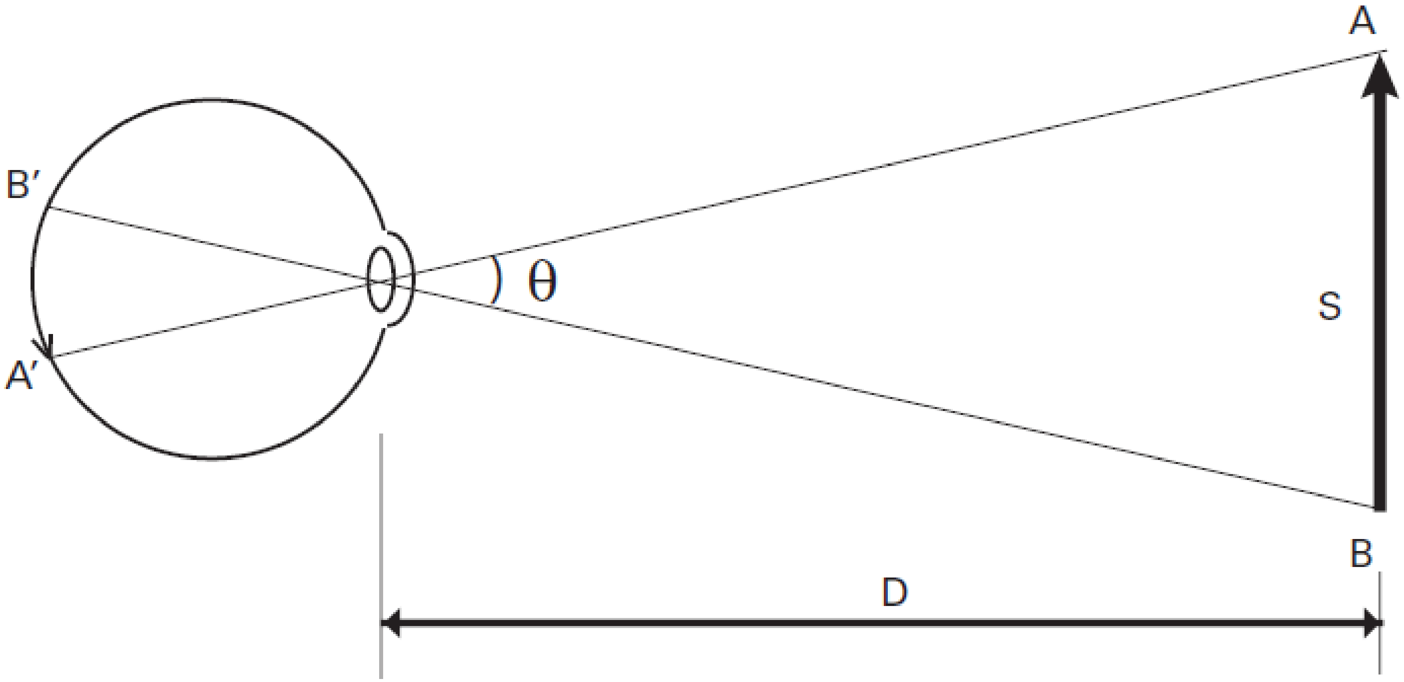
```
VBLtime = Screen('Flip',windowPtr [, when] [,dontclear])
```

Default is to flip now, i.e. at the next VBL interval. However, you can specify a time in the future for the flip to take place. Flip will wait until that time and then flip at the next VBL interval after the specified time.

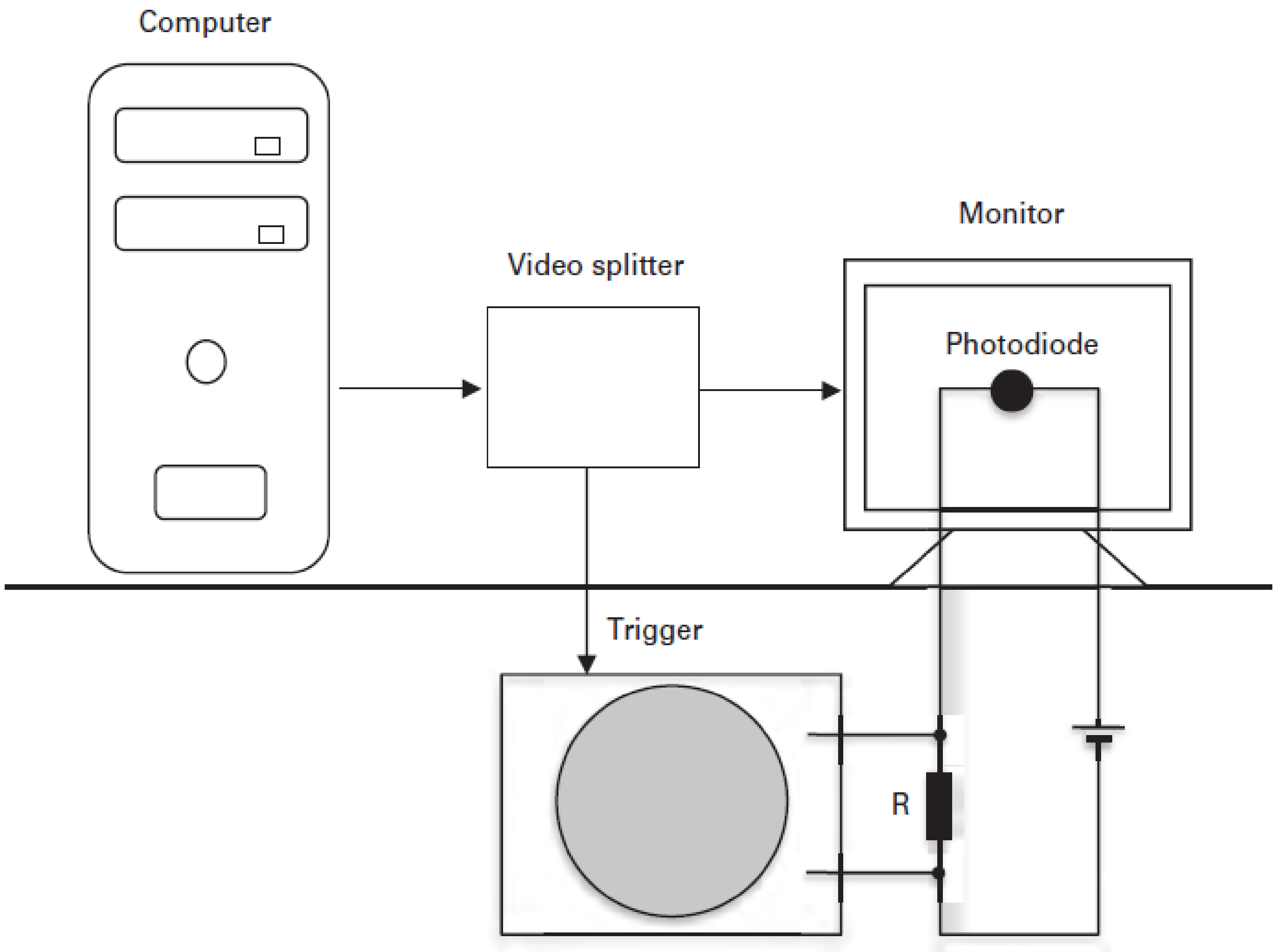
Default is to clear the back buffer. However in some cases you may want to leave the back buffer as is. Default is 0, set to 1 if you don't want it to clear.

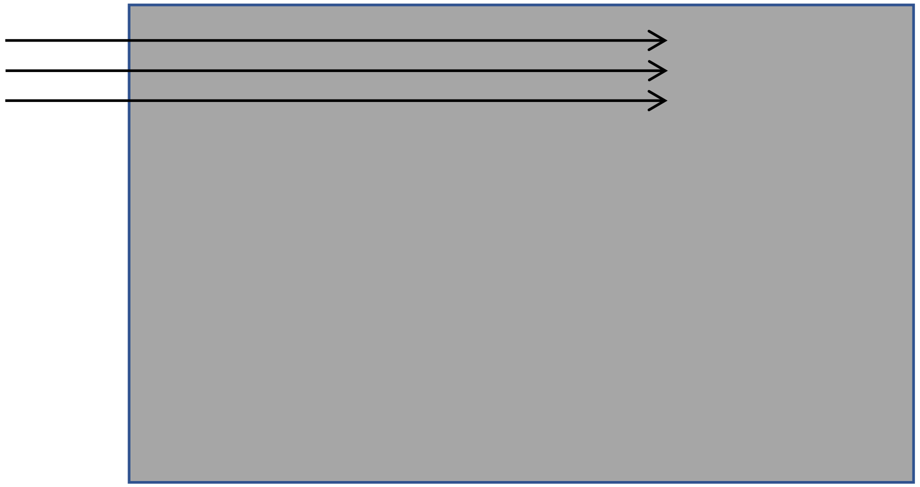
Visual angle

$$\text{Visual angle} = 2\arctan[S/(2D)]$$
$$\approx 57.3\frac{S}{D}\ \text{(degrees), if } S < D.$$



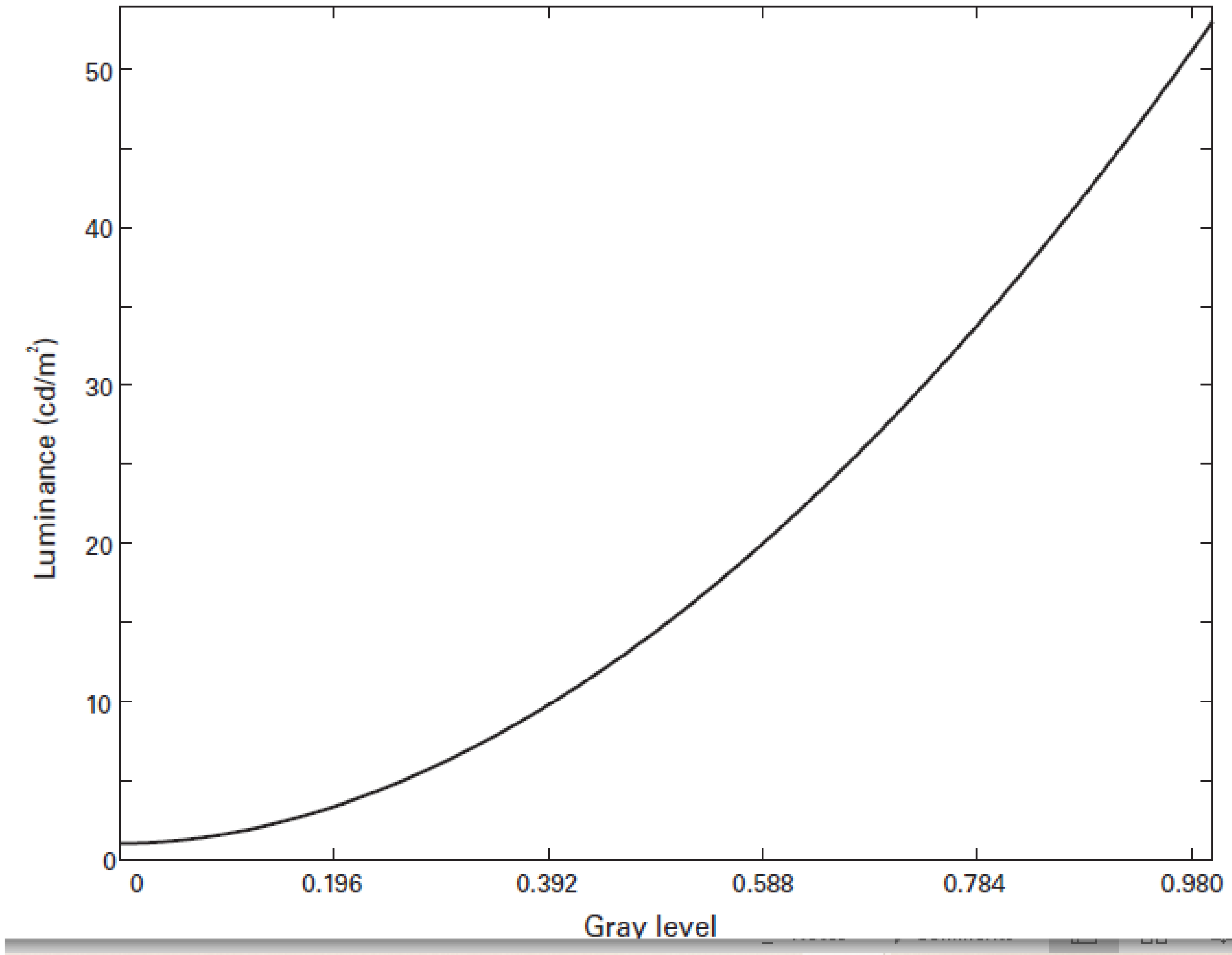
Temporal property of display



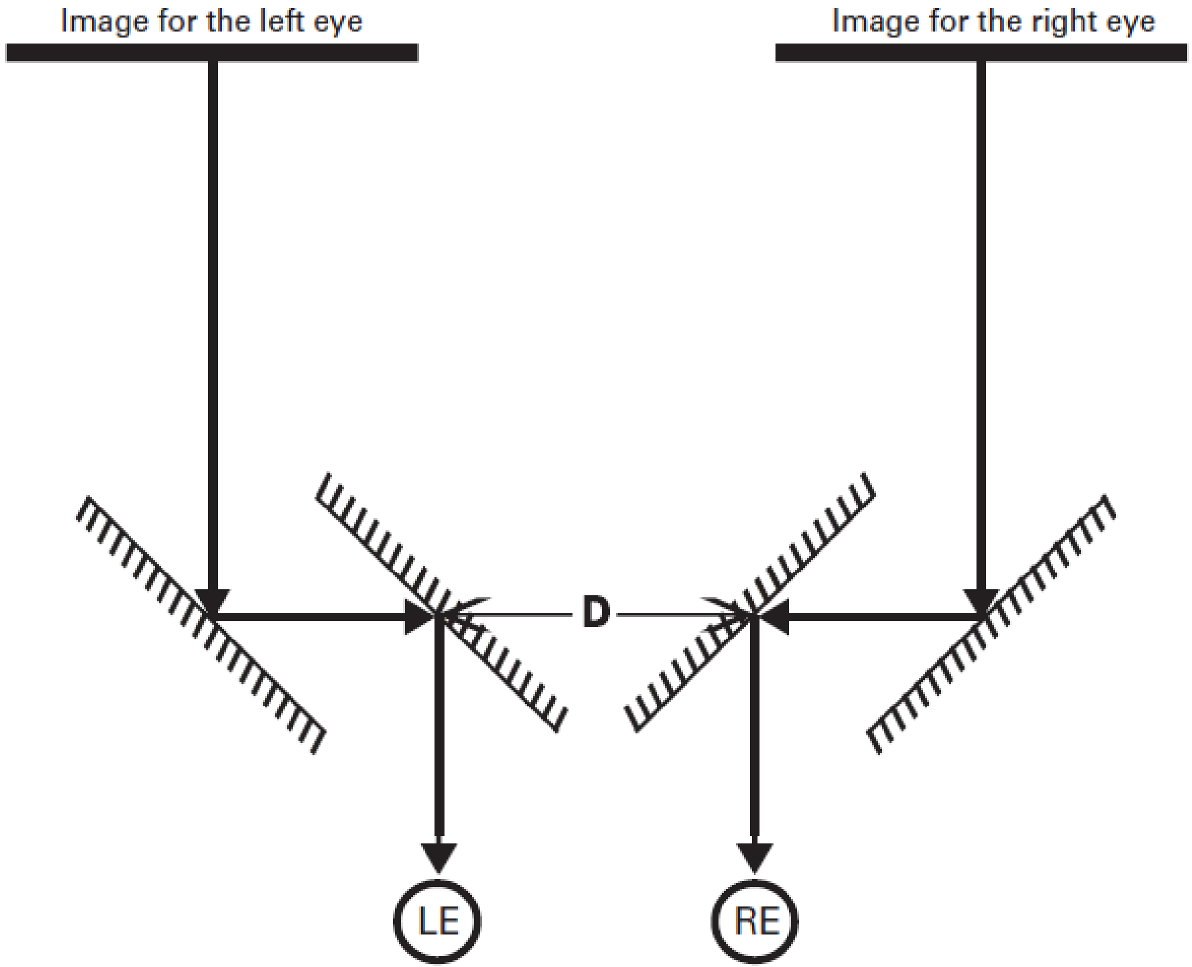


Gamma correction photometer

$$L(U) = L_{\min} + (L_{\max} - L_{\min}) \times U^{\gamma} , \tag{5.1}$$



Different input to each eye



Session #6 assignment

- Install Psychtoolbox
- Test it to work