PsychoPy

Psychology software in Python

Jonathan Peirce

Nottingham Visual Neuroscience
University of Nottingham

Plan

- Roots of PsychoPy
- Design and philosophy
- The use of OpenGL techniques for real-time rendering
- Using PsychoPy for making movies
- Other packages
- Current issues and plans

- Something completely different…?

Roots of PsychoPy

- 1985: My first 'original' BASIC program
- 2000-2003: Python was becoming a viable alternative to Matlab™
  - Psychtoolbox extremely popular
- 2002: Python/OpenGL engine developed as a proof of concept
- 2003: PsychoPy developed more fully
- Now PsychoPy is at 0.93.6 and;
  - downloaded 3000 times
  - around 600 unique visitors a month
  - the main psychophysics software in a number of labs

Some Psychtoolbox code

```
window = Screen(0, 'OpenWindow'); %open a window on screen 0
white = WhiteIndex(window); % pixel value for white
black = BlackIndex(window); % pixel value for black
gray = (white+black)/2;
inc = white-gray;
Screen(window, 'FillRect', gray);

[x,y] = meshgrid(-100:100, -100:100);
m = exp(-((x/50).^2)-((y/50).^2)) .* sin(0.03*2*pi*x);

Screen(window, 'PutImage', gray+inc*m);
Screen(window, 'Flip');

KbWait;
Screen('CloseAll');
```

Aims/Philosophy

- PsychoPy *aims* to be
  - able to generate all stimuli in real time
  - fully platform-independent
  - as far as possible Python-based
  - as user-friendly as possible (especially for non-programmers)
  - collaborative (e.g. open-source)
  - entirely free

  - **a genuine workable alternative to psychtoolbox**

Equivalent PsychoPy code

```
from psychopy import visual, event, core
win = visual.Window([400,400], rgb=[0,0,0])

gabor = visual.PatchStim(win, tex='sin',sf=3,mask='gauss')

gabor.draw()
win.update()

event.waitKeys()
core.quit()
```

Design

- Based on
  - pygame/PyOpenGL (but soon moving to pyglet)
  - numpy
  - wx
- Not highly 'optimised'
- *Fairly* object-oriented
- Interface
  - originally simply script-based
  - now has its own (moderately-featured) IDE
  - will hopefully get a visual drag-and-drop interface

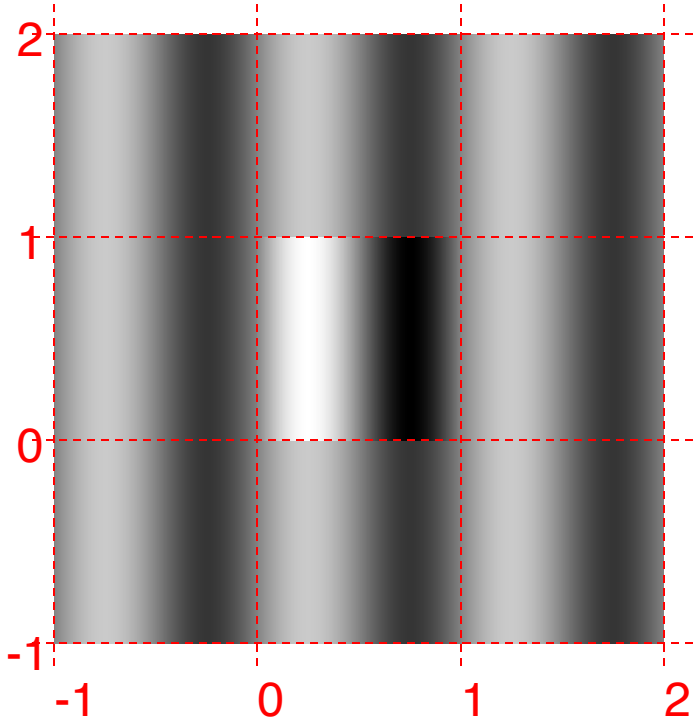
Real-time stimuli

- use hardware-accelerated graphics wherever possible
  - textures, multitextures
  - fragment shaders
- minimize data-transfer between CPU and GPU
  - upload textures in advance of use
- if using an interpreted language you need to minimise number of calls (reduce loops)
  - vertex arrays

PsychoPy
Psychology software in Python

Textures

- `PatchStim` relies heavily on texturing



glBegin(GL_QUADS)

glMultiTexCoord2fARB(*top-left texture coords*)
glVertex3f(*top-left vertex coords*)

glMultiTexCoord2fARB(*bot-left texture coords*)
glVertex3f(*bot-left vertex coords*)

glMultiTexCoord2fARB(*top-right texture coords*)
glVertex3f(*top-right vertex coords*)

glMultiTexCoord2fARB(*top-left texture coords*)
glVertex3f(*top-left vertex coords*)

glEnd()

Alpha masks

We can use a second texture to define an alpha mask (with an independent set of coordinates)

textures

glBegin(GL_QUADS)

glMultiTexCoord2fARB(*texture top-left coords*)
glMultiTexCoord2fARB(*mask top-left coords*)
glVertex3f(*top-left vertex coords*)

glMultiTexCoord2fARB(*texture bot-left coords*)
glMultiTexCoord2fARB(*mask bot-left coords*)
glVertex3f(*bot-left vertex coords*)

glMultiTexCoord2fARB(*texture top-right coords*)
glMultiTexCoord2fARB(*mask top-right coords*)
glVertex3f(*top-right vertex coords*)

glMultiTexCoord2fARB(*texture top-left coords*)
glMultiTexCoord2fARB(*mask top-left coords*)
glVertex3f(*top-left vertex coords*)

glEnd()

PsychoPy
Psychology software in Python

Other uses of textures

- Textures (and therefore `visual.PatchStim`) also have an alpha setting for the stimulus (called opacity)
- Textures and masks can be
  - standard forms ('sin', 'sqr', 'gauss'…)
  - numpy arrays
  - images (anything PIL can load)
- Rotate/translate/scaling are determined at the beginning of drawing each object

alpha, face

PsychoPy
Psychology software in Python

Vertex arrays

- `visual.PatchStim` is powerful enough to cover many vision experiments
- A lot more people work on motion and want to draw large arrays of dots
- `visual.DotStim` handles this case
  - need to avoid looping
  - dot X,Y calculated using numpy array maths
  - OpenGL supports arrays of vertices (and potentially texture coords)

dots

Hardware-specific optimisations

- fragment shaders can be used to accelerate certain aspects of drawing

- frame buffers *will be* used to allow higher-precision imaging (currently 8-bit frame buffer)

3D stimuli

- Can be easily extended (because of OpenGL) to handle a 3D scene with
  - perspective
  - lighting
  - fog(?!)
  - …
- …but that won't get done until someone needs it!

Making movies

- Added facility to generate demo movies for experiments
- `visual.Window` has 3 relevant attributes;
  - `movieFrames` is a list of movie frames (numpy arrays)
  - `getMovieFrame()` appends current frame
  - `saveMovieFrames(f)` outputs current frame list to file (supports tiff, jpg…, gif, mpg depending on platform)

Other stimuli

- PsychoPy is a full-featured system for neuroscience
  - sound stimuli (wavs, numpy arrays)
  - text stimuli (anti-aliased TTFs, with unicode support)
- Integrates easily with other hardware
  - CRS Bits++ (for 14bit DACs and LUTs)
  - parallel, serial, USB ports
  - joysticks (through pygame)
- Routines for calibration
- Routines to help run experiments (e.g. staircase methods)
- Routines for data analysis (bootstrapping and curve fitting)

Comparison with other options

- e-Prime & Presentation:
  - ✗ proprietary, expensive
  - ✗ rely on importing pre-made movies
  - ✓ easy to use (e-Prime) and precise (Presentation)
- Psychtoolbox
  - ✗ based on Matlab™ (and even uglier than most Matlab!)
  - ✓ stable and with large user base
- VisionEgg
  - ✓ Andrew Straw is an excellent programmer
  - ✓ entirely free
  - ✗ not *so* intuitive

Some problems

- Not an *application* like matlab
  - Many users expect to find it in the >Start>Programs menu
  - Or they will double-click a script but they don't see error messages
  - They struggle with the idea of editing code in *anything*

2. Python is a bit of a moving target
   - incompatible dependencies
   - unstable libraries (including PsychoPy itself)
3. My own lack of testing

Some solutions

- built my own editor (PsychoPy IDE)
  - fairly easy to do (love wx and stc)
  - including syntax colouring, folding, auto-complete (rough but works)
  - lightweight way to run scripts and keep output visible
- distribute a folder of dependencies that (should) work together
- hire a professional programmer!

PsychoPy
Psychology software in Python

Future plans

- Most things get implemented when my own work needs them!
- Currently trying to fund a full-time programmer to push things faster than that:
  - distribute PsychoPy as an application (improved IDE and package all dependencies)
  - add a GUI drag-and-drop layer to reduce script-writing for novices
  - debugging and optimising

Something completely different…

- Python-based model of V1 cells that is;
  - functional
  - nonlinear filter-based model;
    - simple/complex continuum
    - 'tuned' suppression (surround suppression)
    - 'untuned' suppression (contrast gain, cross-ori-suppression)
- …but does not;
  - require a whole bank or sheet of neurons
  - mimic the *mechanics* of the cell