Keith Schneider, 17 August 2006

Matlab and Psychophysics Toolbox Seminar

Part 5. Animation II

Rotations

One of the parameters in the `Screen('DrawTexture')` that we haven't utilized yet is the rotation parameter. The video hardware can quickly rotate, translate and resize any image, and the function takes advantage of that. Here's an example of a spinning image (you can try this out with red/green 3D glasses if you have them to experience binocular rivalry).

```
% rotating binocular rivalry stimulus
% Key presses start rotation, stop rotation and exit.
% Wear red/green glasses and fixate at the center.
cycles = 20;    % spatial frequency (cycles per stimulus)
period = 1;     % rotation period (sec)

[w, rect] = Screen('OpenWindow', 0, [0 0 0]);
hidecursor;
xc = rect(3)/2; % coordinates of screen center
yc = rect(4)/2;

% make stimulus image
xysize = rect(4);
xylim = 2*pi*cycles;
[x,y] = meshgrid(-xylim:2*xylim/(xysize-1):xylim, ...
    -xylim:2*xylim/(xysize-1):xylim);
img = zeros(xysize,xysize,3);
circle = x.^2 + y.^2 <= xylim^2;    % circular boundry
img(:,:,1) = (sin(x)+1)/2*255 .* circle;   % red channel
img(:,:,2) = (sin(y)+1)/2*255 .* circle;   % green channel

t = Screen('MakeTexture', w, img);
Screen('DrawTexture', w, t);
Screen('FillOval',w,[255 255 255],[xc-4 yc-4 xc+4 yc+4]); %fixation pt.
Screen('Flip', w);

KbWait;
while KbCheck; end  % make sure all keys are released

% animation loop
keyisdown = 0;
start_time = GetSecs;
while (~keyisdown)
    th = mod(360*(GetSecs-start_time)/period,360); % rotation angle
    Screen('DrawTexture', w, t, [], [], th);
    Screen('FillOval', w, [255 255 255], [xc-4 yc-4 xc+4 yc+4]);
    Screen('Flip', w);
    keyisdown = KbCheck;
end
while KbCheck; end
KbWait;
ShowCursor;
Screen('Close',w);
```

Next we will modify the script we wrote last week to move an image with the mouse. Today we will add in image rotations controlled by the left- and right-arrow keys.

```
img=imread('myimage.jpg');
[yimg,ximg,z]=size(img);
rot_spd = 10;   % rotation speed (degrees per frame)
larrow = KbName('LeftArrow');  % modify this for Windows?
rarrow = KbName('RightArrow');
[w,rect]=Screen('OpenWindow',0,[0 0 0]);
sx = 400;           % desired x-size of image (pixels)
sy = yimg*sx/ximg; % desired y-size--keep proportional
t = Screen('MakeTexture',w,img);
bdown=0;
th = 0; % initial rotation angle (degrees)
HideCursor
while(~any(bdown))  % exit loop if mouse button is pressed
    [x,y,bdown]=GetMouse;

    [keyisdown,secs,keycode] = KbCheck;
    if(keycode(larrow))
        th = th - rot_spd;  % rotate counterclockwise
    end
    if(keycode(rarrow))
        th = th + rot_spd;  % rotate clockwise
    end

    destrect=[x-sx/2,y-sy/2,x+sx/2,y+sy/2];
    Screen('DrawTexture',w,t,[],destrect,th);
    Screen('Flip',w);
end
Screen('Close',w)
ShowCursor
```

Flickering checkerboard

Flickering checkerboard stimuli are commonly used in fMRI experiments to map the visual field. In the following example, the flickering is accomplished by rapidly switching between two complementary images. Also, the apparent rotation of the hemifield stimulus is accomplished by first drawing the fullfield circular stimulus and then drawing a polygon with the background color on top of it to mask half. For a real experiment you would want to add in a fixation point and specify a fixed duration, rather than exiting upon a key press. The most complicated part of this program is the math used to generate the stimulus pattern. The image `checks` that is created contains three values, one corresponding to the background color, and one each for the black and white checkers.

```
% rotating hemifield flickering checkerboard
rcycles = 8;     % number of white/black circle pairs
tcycles = 24;    % number of white/black angular segment pairs (integer)
flicker_freq = 4;    % full cycle flicker frequency (Hz)
flick_dur = 1/flicker_freq/2;
period = 30;     % rotation period (sec)

[w, rect] = SCREEN('OpenWindow', 0, 128);
HideCursor
xc = rect(3)/2;
yc = rect(4)/2;

% make stimulus
hi_index=255;
lo_index=0;
bg_index =128;
xysize = rect(4);
s = xysize/sqrt(2); % size used for mask
xylim = 2*pi*rcycles;
[x,y] = meshgrid(-xylim:2*xylim/(xysize-1):xylim, - ...
    xylim:2*xylim/(xysize-1):xylim);
at = atan2(y,x);
checks = ((1+sign(sin(at*tcycles)+eps) .* ...
    sign(sin(sqrt(x.^2+y.^2))))/2) * (hi_index-lo_index) + lo_index;
circle = x.^2 + y.^2 <= xylim^2;
checks = circle .* checks + bg_index * ~circle;
t(1) = SCREEN('MakeTexture', w, checks);
t(2) = SCREEN('MakeTexture', w, hi_index - checks); % reversed contrast

flick = 1;
flick_time = 0;
start_time = GetSecs;

while (1) % animation loop
    thetime = GetSecs - start_time; % time (sec) since loop started
    if thetime > flick_time     % time to reverse contrast?
        flick_time = flick_time + flick_dur; % set next flicker time
        flick = 3 - flick;
    end
    SCREEN('DrawTexture', w, t(flick));

    % draw mask
    theta = 2*pi * mod(thetime, period)/period;
    st = sin(theta);
    ct = cos(theta);
    xy = s * [-st,-ct; -st-ct,st-ct; st-ct,st+ct; st,ct] + ...
        ones(4,1) * [xc yc];
    Screen('FillPoly', w, bg_index, xy);
    SCREEN('Flip', w);

    if KbCheck
        break % exit loop upon key press
    end
end
ShowCursor
SCREEN('Close',w);
```

Movies

Another type of animation is a movie. A movie is an animation in which you have pre-computed all of the frames of the animation, storing them in sequence into memory using the `Screen('MakeTexture')` function. The frames can then be played, one frame per screen refresh, using the `Screen('DrawTexture')` function. A movie animation needs to be used when you have imported the images you want to display, or the computer is too slow to generate them on the fly.

For example, let's modify our mouse moving and rotating script to display a random dynamic noise stimulus.

```
nframes = 100;  % number of frames in movie
s=200;  % size of square (pixels)
rot_spd = 10;   % rotation speed (degrees per frame)
larrow = KbName('LeftArrow');  % modified this for Windows?
rarrow = KbName('RightArrow');

[w,rect]=Screen('OpenWindow',0,[0 0 0]);
% compute movie frames
t = zeros(1,nframes);
for i=1:nframes
    img=255*rand(s,s,3);    % colored noise image
    t(i)=Screen('MakeTexture',w,img);
end
i=0;
bdown=0;
th = 0; % initial rotation angle
while(~any(bdown))  % exit loop if mouse button is pressed
    [x,y,bdown]=GetMouse;

    [keyisdown,secs,keycode] = KbCheck;
    if(keycode(larrow))
        th = th - rot_spd;  % rotate counterclockwise
    end
    if(keycode(rarrow))
        th = th + rot_spd;  % rotate clockwise
    end

    destrect=[x-s/2,y-s/2,x+s/2,y+s/2];
    i=mod(i,nframes)+1; % cycle through frames
    Screen('DrawTexture', w, t(i), [], destrect, th)
    Screen('Flip', w);
end
Screen('Close',w)
```

You can also import your own movies. This next example may tax your video card. On Macs the movie playback functions utilize QuickTime. Use your own movie or I can supply one.

```
% translate and rotate movie during playback
rot_spd = 10;   % rotation speed (degrees per frame)
larrow = KbName('LeftArrow'); % modify for Windows?
rarrow = KbName('RightArrow');

[w,rect]=Screen('OpenWindow',0,[0 0 0]);
[m dur fps sx sy] = Screen('OpenMovie', w, 'mymovie.mov');
f=0;
bdown=0;
th = 0; % initial rotation angle
Screen('PlayMovie', m, 1, 1); % start movie playback
while(~any(bdown))  % exit loop if mouse button is pressed
    [x,y,bdown]=GetMouse;

    [keyisdown,secs,keycode] = KbCheck;
    if(keycode(larrow))
        th = th - rot_spd;  % rotate counterclockwise
    end
    if(keycode(rarrow))
        th = th + rot_spd;  % rotate clockwise
    end

    destrect=[x-sx/2,y-sy/2,x+sx/2,y+sy/2];
    t = Screen('GetMovieImage', w, m, 1); % grab frame
    Screen('DrawTexture',w,t,[],destrect,th)
    Screen('Flip',w);
end
Screen('CloseMovie', m);
Screen('Close',w)
```

The way this script works is to grab the current frame into a texture using the `Screen('GetMovieImage')` function. This texture can then be quickly rotated, translated, resized, etc. You could make the movie bounce around, play multiple instances of the movie by drawing the texture in different locations, etc.

As you can see, the Psychophysics Toolbox is quite powerful. For more examples in changing the movie playback rate, rewinding the movie, and so on, see `PlayMoviesDemoOSX.m` and related demos in the `Psychtoolbox/PsychDemos/QuickTimeDemos` directory, or read the help files for the `Screen('OpenMovie')` and `Screen('PlayMovie')` functions.

Dot fields

If you want to display a field containing a large number of moving dots, you may find that the `Screen('FillOval')` is too slow to draw a sufficiently high density of dots each frame. For this purpose, use the `Screen('DrawDots')` function. Let's modify our translating and rotating script to display a field of moving dots:

```
dot_speed = 10; % dot speed (pixels/frame)
ndots = 500;    % number of dots
dot_w = 10;     % width of dot (pixels)
s = 500;        % field size (pixels)
rot_spd = 2;    % rotation speed (degrees per frame)
larrow = KbName('LeftArrow'); % modify for Windows?
rarrow = KbName('RightArrow');
x = s * rand(1,ndots);  % initial position
y = s * rand(1,ndots);
w=Screen('OpenWindow',0,0);
Priority(MaxPriority(w));
% Enable alpha blending for smoothed points:
Screen('BlendFunction', w, GL_SRC_ALPHA, ...
    GL_ONE_MINUS_SRC_ALPHA);
bdown=0;
th = 0; % initial heading
HideCursor
while(~any(bdown))  % exit loop if mouse button is pressed
    [mx,my,bdown]=GetMouse;
    [keyisdown,secs,keycode] = KbCheck;
    if(keycode(larrow))
        th = th - rot_spd; %rotate heading counterclockwise
    end
    if(keycode(rarrow))
        th = th + rot_spd; %rotate heading clockwise
    end
    dx = dot_speed * cos(th*pi/180); % x-velocity
    dy = dot_speed * sin(th*pi/180); % y-velocity
    x = mod(x+dx,s); % update positions, looping at edges
    y = mod(y+dy,s);
    Screen('DrawDots',w,[x;y], dot_w, 255, [mx my]-s/2, 1);
    Screen('Flip',w);
end
Screen('Close',w)
ShowCursor
Priority(0);
```

Also, see the `DotDemo.m` program for an example with multiple dot sizes and colors. If your video card doesn't support alpha blending, you will see square dots instead of circular ones.