|  |  |
| --- | --- |
| DPX: an Experiment Preparation System USER MANUAL | C:\Users\jacob\AppData\Local\Microsoft\Windows\INetCache\Content.Word\dpxLogo.png |

# Contents

1. Introduction
2. Installing and maintaining DPX
3. Creating and running experiments
4. DPXD and writing analyses
5. Overview of main stimuli and response classes
6. Creating additional stimuli and response classes

# Introduction

DPX is a Matlab extension to the Psychtoolbox to prepare, run, and analyze psychophysical experiments for vision research. DPX stands for "Duijnhouwer Psychtoolbox eXperiments". This title reveals the origin of DPX as an (online) repository that I created to bring together and share the Psychtoolbox-based experiment scripts that were in use at the laboratory I worked in at the time.

These scripts were typically created by making a copy of a previous Psychtoolbox-based experiment that looked most like the one that was in mind for the next. This was then hacked and tweaked until it approximated the newly intended purpose. Obviously, over time, this practice led to obfuscated spaghetti code that was prone to bugs, a pain to maintain and adapt, and generally a waste of time.

I realized that psychophysics lends itself perfectly to an object-oriented (OO) approach, a style of programming that is well implemented in Matlab since edition 2008a. In this manual, I assume familiarity with the basic Matlab language. In the OO-approach, in short, any conceptual unit of the problem at hand is represented as an object that is defined by a class that has certain properties (variables it has) and methods (things it can do, similar to functions in standard Matlab).

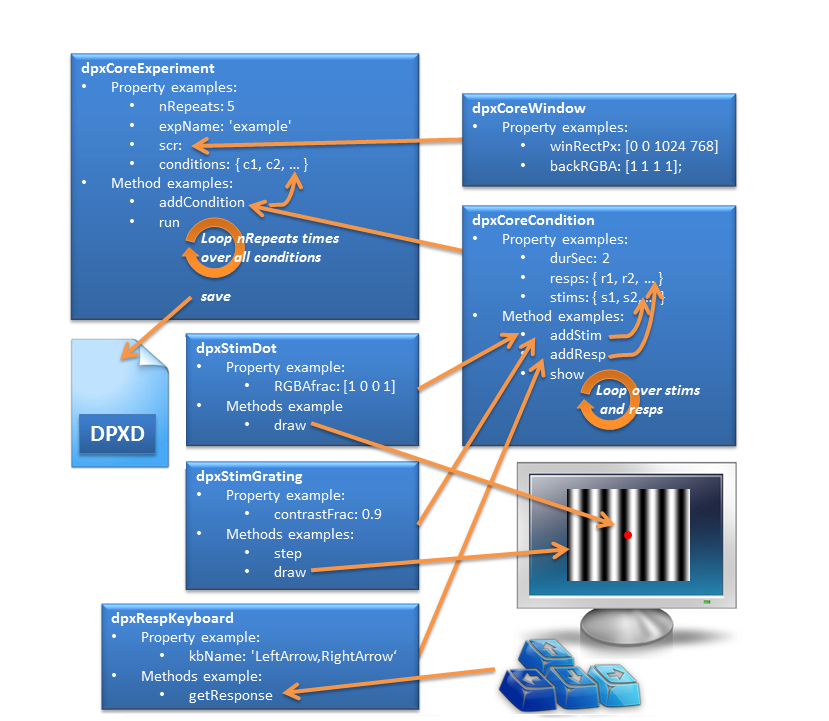
The main conceptual units that comprise a vision-science experiment are the experiment itself, the display-hardware, the conditions, the stimuli, and the responses. These are neatly hierarchically related (Figure 1). The experiment-class (dpxCoreExperiment), at the top, contains a display-object (class dpxCoreWindow) and a cell-array to hold condition-objects (class dpxCoreCondition). The condition objects, in turn, contain a number of stimulus-objects (dpxStim) for output and one or more response-objects (dpxResp) to register the input of the observer. Examples of methods of dpxCoreExperiment are functions to add condition-objects (addCondition) and to start the experiment (run). The dpxCoreExperiment class also takes care of saving the data, which is implemented as a hidden-method called at the end of the run method. Examples of properties of the stimulus class are the horizontal (xDeg) and vertical (yDeg) position in degrees relative to the center of the screen. More examples of methods and properties that are pertinent to the different levels of the hierarchy are indicated in Figure 1.

Figure Outline of the object-oriented structure of DPX

Over the years, the OO-approach of DPX has proved an efficient way to perform research by promoting code reuse and consistency. Using DPX reduces the time and effort needed to develop an experiment considerably. In addition, because it produces a simple, standardized, and easy to work with data format (called DPXD, see Section XXX) the creation of analyses is also fast and future-proof.

In the following section, I will explain how to obtain, install, and maintain DPX on your computer using Apache Subversion ([SVN](https://en.wikipedia.org/wiki/Apache_Subversion)). Then I will explain, using examples included in DPX, how to run and create experiments (Section 3). Many stimulus and response classes are included in DPX, and the number is ever growing, so it may well be possible that you can simply create your experiment by combining these existing classes. In Section 4, however, I will explain how to create additional ones. Finally, in Section 5, I will demonstrate how to create analyses using the DPXD format and its companying functions.

# Installing and maintaining DPX

# Creating and running experiments

A DPX experiment is defined in a Matlab function. The code of a simple “Hello, World!”-example is shown below. All examples covered in this manual (and more) can be found in the folder DPX\dpxExamples\. Run this example by typing dpxExampleHelloWorld in the Command Window followed by Enter or view it using edit dpxExampleHelloWorld.

|  |  |
| --- | --- |
| TIP | Type dpxEx followed by Tab to bring up a list of all DPX examples. |

|  |
| --- |
| function dpxExampleHelloWorld    % function dpxExampleHelloWorld  % Example Hello World experiment in a small window.  % Jacob Duijnhouwer, 2015-10-19    % Instantiate the experiment object E  E=dpxCoreExperiment;  % Define the expName property, this is typically the filename  E.expName='dpxExampleHelloWorld'; % alternative E.expName=mfilename;  % Set a few display properties of the dpxCoreWindow object within E.  E.scr.winRectPx=[20 20 640 480]; % Set the display area.  E.scr.skipSyncTests=true; % For this example, skip Psychtoolbox testing    % Make two conditions, one to say 'Hello,' the other to say 'World!'  for conditionNumber=1:2  % Instantiate a condition object  C=dpxCoreCondition;  C.durSec=2; % Make the condition last for two seconds  % Instantiate a text stimulus object  T=dpxStimTextSimple;  if conditionNumber==1  % Set character string for condition 1  T.str='Hello,';  else  % Set character string for condition 2  T.str='World!';  end  % Add the stimulus to the condition  C.addStim(T);  % Add the condition to the experiment  E.addCondition(C);  end    % Repeat each condition once (that is, present one block)  E.nRepeats=1;  % Show in not-shuffled order  E.conditionSequence='notShuffled'; % default: 'shufflePerBlock'  % Run the experiment ...  E.run;  end |