DMDX Introductory Tutorial

Isabelle Darcy, Fall 2010

[updated August 2014]

This tutorial is based on available online tutorials, in particular, on that by Matt Davis

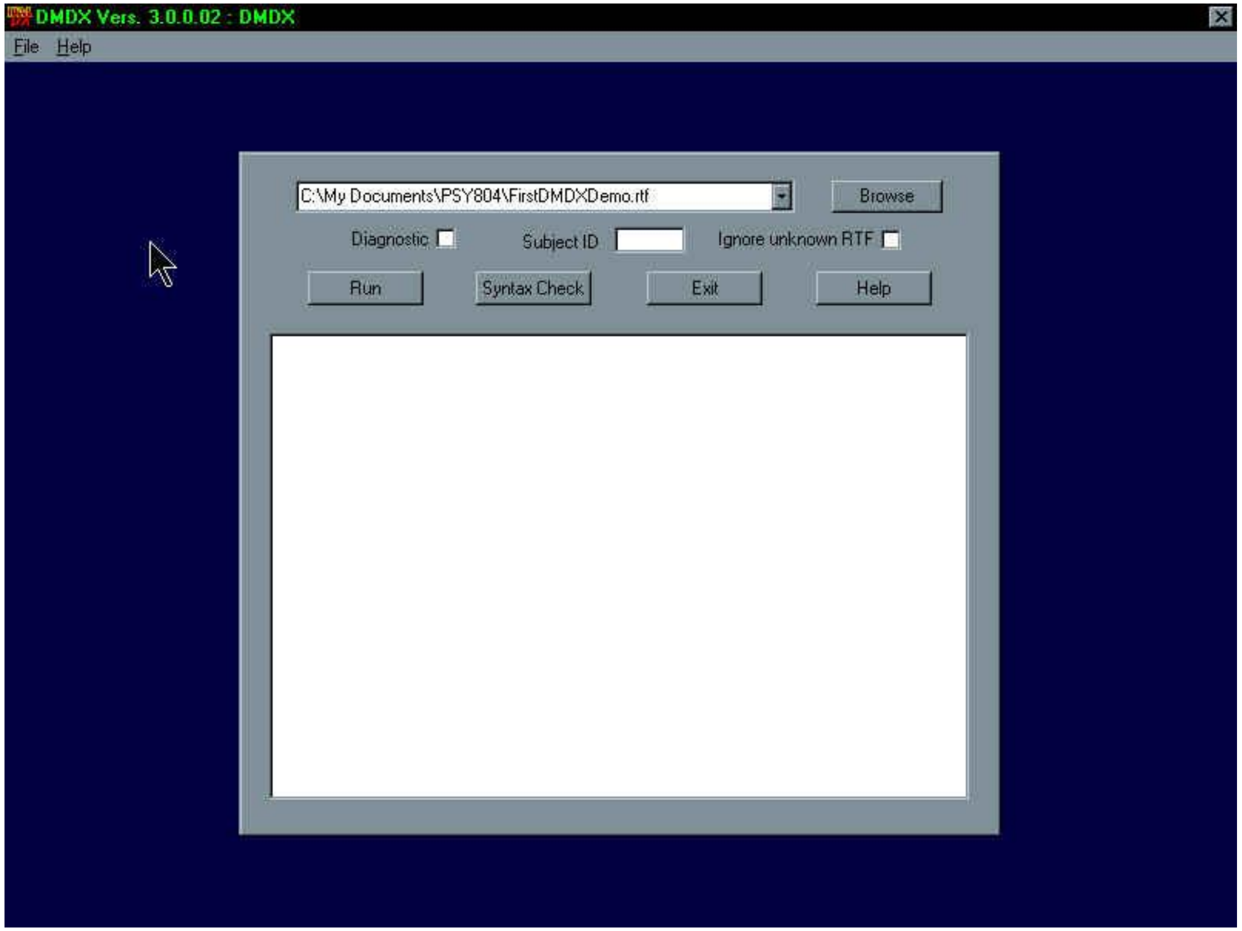
What is DMDX ?

DMDX is your friend ☺

What is DMDX?

- DMDX is:
  - Reliable
  - Flexible
  - Usable for non-programmers
  - Runs on modern personal computers under Windows 95/98, Windows XP (runs great), Windows 7 (ok so far). We’ve had problems with Windows Vista.
  - Free!

Introduction to DMDX (and other stuff)



http://dionysus.psych.wisc.edu/methods/DMDX/DMDXHelp.htm

Note: Some examples and content are from a tutorial originally created by Matt Davis at the University of Cambridge

Basic Info

- DMDX is software for the experimental control and timing of stimulus display
- DMDX was created by Jonathan and Ken Forster in the Department of Psychology at the University of Arizona
- It can be downloaded from the DMDX website at:
  - http://www.u.arizona.edu/~kforster/dmdx/dmdx.htm

- User support is managed through an email listserv [updated 2014]:
  - Send a message to list@list.arizona.edu from the address you want to subscribe to the list.
  - In the subject line of your message, type in: **subscribe dmdx Firstname Name** (replace 'Firstname Name' by your own first name and name).
  - Once your subscription is confirmed posts can be made to dmdx@list.arizona.edu. Please don't send attachments to the list.

    Your subscription should also be manageable by pointing your browser at https://list.arizona.edu/sympa/info/dmdx.

Getting DMDX up and running

- DMDX needs to be set up properly, using associated program TIMEDX
- Each configuration will be specific to each computer/monitor
- Each time you want to use a script on a different machine, you may need to adjust the script and/or configuration of the machine.

- We'll now see how to configure properly DMDX

- The steps in using TimeDX are as follows:
- For each **video mode** that you will be using:
  - *Check* **Video Mode**
    Check that the Video Mode you want to use works. Depending on how good your system is, some modes may not work.
  - *Time* **Refresh Interval** *for video mode*
    This is a technical property DMDX needs to know if it's going to work properly. Leave it timing for twenty seconds.
  - *Check Video timer with Millisecond timer*
    (Advanced -> "Video and Millisecond timer")
    Check that the millisecond clock and the system clock keep pace with each other (first two lines).

- *Synchronise DMDX with video card*
  (Advanced -> “Vertical Retrace Sync Thread”)
  Click on “Do Test” and then “Save last used values in registry”.

- Now set up everything else.
  - Check your soundcard will work using the **Sound** button.
  - Check DMDX can detect **Input**. For now, just check the keyboard. An error will come up if something is amiss.
- This is the “quick fix” way of setting up DMDX. Depending on your application, you may need to get involved in more technical steps.

How to run an experiment with DMDX

Based on tutorials by Arie van der Lugt & Matt Davis

What is DMDX?

DMDX is a program that can be used to run psychological experiments

DMDX presents lists of stimuli (*pictures*, *text*, *sounds)* that are prepared and specified by you.

DMDX collects responses
For example: Which keys are pressed on the keyboard and how long did it take before they were pressed

DMDX saves the RT and correct/error responses in a data file for later analysis

Running an experiment with DMDX

Stimulus Materials

- Text (in the script file)
- Pictures (.bmp files)
- Sounds (.wav files)

The exerimental "script"
(or "item file")
(specifies how and when stimuli will be presented, how and when responses are recorded)
created as a text file,
saved in .rtf format

Data file:

Contains RT and error codes for each trial

Saved as a text file with the extension .azk

How does DMDX work? The item file (or “script”)

- A simple text file, saved in Rich Text Format (***.rtf***), that contain a sequence of instructions telling DMDX

  1) What stimuli to present to subjects

  2) How long to present them for

  3) When to start measuring response times

  4) What response to expect (“positive”/”negative”)

- Item files are divided into two sections
  – the header line
  – A list of item lines (= trials)
- The header line specifies certain global parameters -- *e.g. background colour of the screen, default duration of interval between trials, which keys to use as response keys*
- Each item line tells DMDX what to display on that trial.
- *Note:* instruction and between-block displays are created as pseudo-trials.

How does DMDX work? The stimulus files (or “stimuli”)

Stimulus Files

- These are files that contain the stimuli that you want DMDX to display

  *1) Speech Files (**.wav**)*

  *2) Picture Files (**.bmp**)*

  3) Movies (.mov)

- NB: Written words are typed directly in the item file

(i) What's in an item file?

- Each and every item file has to start with a header line. This sets various parameters for the experiment. An example is shown below:

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

This **looks** scary, but isn't when broken down piece by piece.

A few of these keywords will go in more or less every file that you'll use. I'll explain most of these, but there's some more documentation in the DMDX help files.

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

azk Output data to an Ascii text file

DMASTR used to use completely nightmarish data files. Now everything is a **lot** simpler and the output files are a lot easier to work with.

DMDX data files all have the suffix **.azk**

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

cr Continuous Running

Without this, DMDX will pause after each item and wait for the subject to request the next item. This may be useful for working with patients, but most experiments tend to be continuous running with each item being played after the other

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

fd Standard Frame Duration measured in ticks

This tells DMDX how long to display each stimulus word or picture for

d Delay from end of one item to the start of the next item, again measured in ticks

So what the heck is a tick? Simple answer:

11.80 milliseconds on the desktop machines in the testing rooms

16.67 milliseconds on the portable machines

So what the heck is a tick? Correct answer:

- A tick is the unit used by DMDX to time screen displays

  and

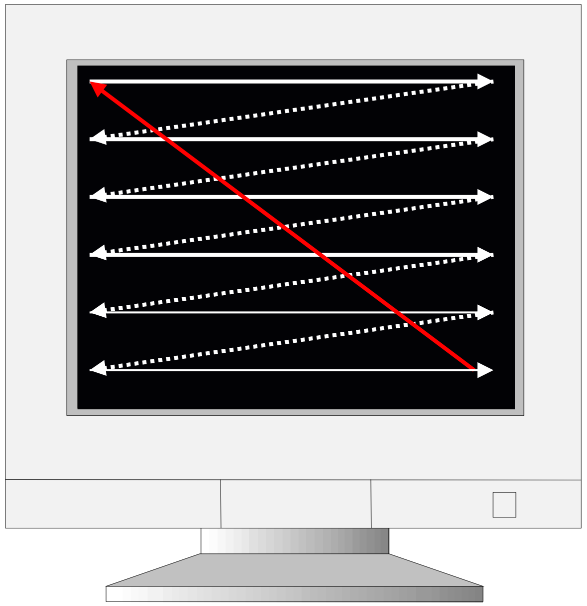
- The time it takes the cathode ray on the monitor to scan the tube

- **NB: This can change at different times for different machines**

- You can find out how long a tick is using a program TimeDX

  or

- By looking in the top line of an .azk data file



Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

t Time-out measured in milliseconds

This tells DMDX how long to wait for subjects to make a response. If no response has been made before this time then the program moves on to the next item.

A value of about 2.5 seconds is about right for normal adults. Elderly volunteers, children or brain-injured patients may need more time.

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

vm Video Mode.

This can affect the length of a tick so you should use this value unless you have a good reason not to. Video mode is usually determined by running TimeDX.

Video mode is comprised of 5 values that depend on your system. The first three are roughly the size of your monitor screen in pixels, and the last two are about the color definition and the refresh rate of the monitor.

Another example: [vm]

```
<ep> <cr> <fd 1> <d 119> <t 1500> <id keyboard> <dbc
210210210> <dfs 36> <dwc 0> <vm 1024,768,768,16,60> <nfb>
<rcot><eop>
```

vm Video Mode. Video modes are defined as follows:

1024	number of pixels horizontally,
768	number of pixels vertically
768	number of scan lines
16	number of bits of color
60	refresh rate (available on XP only)
	[if you don't run Windows XP, this value will likely be 0]

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

id Identifiers for the method used to record responses

PIO12	The response box is to be used
keyboard	The keyboard is to be used

We include both methods, since it is often useful to have the keyboard available when setting up a file

Another example [id]

```
<ep> <cr> <fd 1> <d 119> <t 1500> <id keyboard> <dbc 210210210> <dfs 36> <dwc 0> <vm 1024,768,768,16,60> <nfb> <rcot><eop>
```

id Identify Device

This tells DMDX to monitor various devices for responses during execution of the item file.

PIO12	The IO card (buttons attached to this)
keyboard	The keyboard
mouse	The mouse
digitalVOX	A digital voice activated switch
RecordVocal	Saves verbal responses as .wav files

In general, you should not use keyboard for recording timed responses if possible. Prefer a response box. However, it is useful to start and pause the item file.

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

nfb No FeedBack. Don't provide feedback on subjects response times and errors.

Without this option, DMDX will display messages like:

Correct 550	for a correct response with an RT of 550ms
Wrong	for an incorrect response

This is too distracting for most of the tasks we use, but may be useful in some circumstances.

Headlines

```
<azk> <cr> <fd 17> <d 169> <t 2500>
<vm 1024, 768, 768, 16, 0> <id PIO12>
<id keyboard> <nfb> <dbc 255255255> <dwc 0>
```

dbc	Default Background Colour
dwc	Default Writing Colour

Colours are defined by three numbers from 0 to 255 representing the brightness of each gun in the order red, green, blue

Therefore	255000000 is red
	000255000 is green
	255255255 is white
	000000000 is black

Another example [dbc dwc dfs]

```
<ep> <cr> <fd 1> <d 119> <t 1500> <id keyboard> <dbc
210210210> <dfs 36> <dwc 0> <vm 1024,768,768,16,60>
<nfb> <rcot><eop>
```

dbc	Default Background Color [default is white]
dwc	Default Writing Color [default is black]
dfs	Default Font Size

210210210 is a light gray that makes for a nice background.

Another option [rcot]

```
<ep> <cr> <fd 1> <d 119> <t 1500> <id keyboard> <dbc
210210210> <dfs 36> <dwc 0> <vm 1024,768,768,16,60>
<nfb> <rcot><eop>
```

rcot Record Clock On Time Keyword

If you include this in your header, DMDX will include the time of each clockon (*) in your experiment. The first clockon will be set as 0 and all later will be relative to that.

Very helpful for checking timing

Very helpful for fMRI studies

Include it as a rule

(ii) What’s in an item file?

- As well as a header line an item file will also contain a set of lines describing the items being presented
- Here’s some example items from a DMDX file:

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

Again we’ll go through this step by step.

Item lines

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

+ / -	The first character tells DMDX whether to expect a yes (+) or a no (-) response.

Since the task that we're using with these items is lexical decision then words will be preceded by + and non-words by a -

Item lines [+ / -]

```
+11 * "RABIES" <% 30>/;
-22 * "BRANTLY" <% 30>/;
-23 * "SKELVE" <% 30>/;
+14 * "JUMP" <% 30>/;
```

+ / -   The first character tells DMDX whether the "yes" (+) or "no" (-) response is correct for that item

^ indicates that the correct response is no response.

= indicates that any response is correct

By default, the RIGHT and LEFT SHIFT buttons on keyboard are mapped to yes/no respectively if you are using the keyboard

Item lines

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

These numbers are item numbers. You should assign a unique number to every item in your file. Reaction time data will be recorded by these numbers.

Since in analyzing our data we will split the RTs into word and non-word trials our numbers start at 1 for words and at 101 for non-words.

You should use a meaningful coding scheme for selection of item numbers. This will speed up your processing of the behavioral data.

Item lines: the item number or trial code

```
+101001 * “BOTTLE” /;
+102002 * “ZOOLOGICAL” /;
-100103 * “WONKER” /;
-100104 * “PIRAF” /;
```

These numbers are trial codes that will later be used to sort the data. In a psycholinguistic experiment where each item is only presented once, each trial will have a unique code. In other cases there may be a small number of trial types each of which occurs many times.

You should use a sensible coding system that can be used to sort the results per condition afterwards. For example, the 3rd digit is used to code NONWORD/COMMON WORD/ RARE WORD.

An Aside on Item Numbers

- **You should code all of your within subject condition information into your item number.**
- **Then, you can use simple mathematical operations (mod and trunc) in Excel or SPSS to parse an item number into separate within subject variables.**

Trunc returns the integer portion of a real number.
For example: Trunc (1.2) = 1; Trunc (2.7) = 2

We can also put expressions inside the ( )
For example: Trunc(112 / 10) = 11;   Trunc(191 / 100) = 1

Mod returns the integer remainder of a division operation
For example: Mod(5,2) = 1; Mod (11,4) = 3

An Aside on Item Numbers

The basic strategy to extract a target position in a numeric string (e.g., 23543) involves 2 steps.

1. Use Trunc and division to isolate the target position in the "ones" column. Do this by taking the Trunc of the string divided by a factor of 10 (10, 100, 1000) chosen such that the "1" in the factor of ten is in the same position as the target position in the numeric string.
2. Trunc (23543/100) = 235
3. Take the Mod of the result divided by 10
4. Mod(235,10) = 5
5. These two steps can be combined into one formula
6. Mod(Trunc(23543/100),10) = 5

An Aside on Item Numbers

A quick quiz:

1. What is the formula to extract the 7 from 2455471

   Mod(trunc(2455471 / 10),10)

2. What about the 2?

   Mod(trunc(2455471 / 1000000),10)

3. What about the 1?

   Mod(trunc(2455471 / 1),10) or Mod(2455471,10)

4. What about the 55?

   Mod(trunc(2455471 / 1000),100)

**See demo in Excel and SPSS

An Aside on Item Numbers

Another strategy uses the function “text to columns” in Excel:

1. Mark the column with all your item numbers
2. Through the function “text to columns”, choose appropriate delimiters to split that content over several columns.

Item lines

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

* Clock on signal.

This marks the point at which DMDX should start the clock to record reaction times. In this case we will start timing when the text is presented on the screen, in other experiments this may be more complicated.

NB: the ordering of parameters within a frame does not matter (i.e., it does not affect timing, etc).

An aside on the position of the clock-on signal

- /<wav 2> "file" */ --- / *<wav 2> "file" /   --- / * /<wav 2> "file" / -- /<wav 2> "file" / * /
- For testing purpose, I manipulated the sound files (just 3), so all of them are of equal length. I also added a beep (monotone) at the offset of all sound files (the total duration of each = 850ms), so I tried to respond as soon as I heard the beep and also the onset of the sound file (in two different programs.). The results for
- /<wav 2> "file" */ and
- / *<wav 2> "file" /
- were similar: if I responded at beep, RTs would range from 900-960ms (50-110ms plus 850ms), which means that RT was likely to be measured from the onset of the frame,
- Likewise, / * /<wav 2> "file" / yielded similar results to /<wav 2> "file" */ and / *<wav 2> "file"/.
- If I responded as soon as the sound file was coming out (i.e. responded near the half way of the word), the average RT was about 250ms.
- Finally, /<wav 2> "file" / * / yielded the fastest RT when I responded at beep, ranging from 60-105ms (only 3 trials so far). However, if I responded at the onset of the sound file, then RT wouldn't even be collected because the clock hadn't even been turned on.
- It seems that RT is collected at the onset of the frame regardless of the position of the clock-on signal within that frame. I looked up in DMDX help, and there is an example: +004 "The" / "boy" / "went" / * "HOME" / ; followed by a short description:
  "...The subject is intended to respond to the item in upper-case letters, hence the display-control character * (the clock-on symbol) is located in this frame."

Since the example is for visual stimuli presentation, it can't really explain whether RT will be collected at the onset or offset of the frame. (Thanks to Chung-Lin Yang for figuring this out)

Item lines

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

This tells DMDX what text to display on the screen. It will be displayed for the default frame duration that was specified in the header line. By default, DMDX will display the string between the `" "` as text on the screen.

For our file this was 17 ticks. On the desktop machines, a tick is 11.80 ms. Therefore:

$$\mathbf{17 \times 11.80 \approx 200\ milliseconds}$$

Item lines

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

/ This marks the end of a frame. At the this point the screen is cleared (i.e. after the word has been displayed for 200ms)

If we omitted this / then the word would stay on the screen until we next asked DMDX to display something else - in this case until the next line in the file.

Item lines

```
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
```

; This marks the end of an item. At this point DMDX will wait for a response or a time-out and then move on to the next item.

Before displaying the next item DMDX waits for the delay time set in the header line (<d> parameter). (Roughly, this corresponds to the Inter-Trial-Interval).

For our file this was 169 ticks. Therefore:

$$\mathbf{169 \times 11.80 \approx 2000\ milliseconds}$$

Item lines

```
<azk> <cr> <fd 17> <d 169> <t 2500> <vm 1024, 768, 768, 16, 0>
<id PIO12> <id keyboard> <nfb> <dbc 255255255> <dwc 0>
0 "Press SPACEBAR or FOOTPEDAL to start";
+1 * "RABIES" /;
-101 * "BRANTLY" /;
-102 * "SKELVE" /;
+2 * "JUMP" /;
0 "The END! Thank you for taking part.";
```

Having put the header line and the item lines together we can now run an experiment. To do this we have added two lines at the start and end of the experiment.

These use ID 0 to display information on the screen for the subject. There’s no clock on signal, so DMDX will ignore responses and move on when the spacebar is pressed.

Item lines [bmp jpg wav]

```
+11 * <bmp> "RABIES" <% 30>/;
```

<bmp> This parameter tells DMDX to present a bmp file with the file name (excluding extension) specified in " " (i.e., rabies.bmp).

<jpg> does the same for a jpg image

```
+11 * <wav 2> "RABIES" <% 30>/;
```

<wav N> This parameter tells DMDX to play a wav sound file with the file name specified in " " (i.e., rabies.wav). N indicates:

0: left channel
1: right channel
2: both channels (stereo)

Item lines [%]

```
+11 * "RABIES" <% 30>/;
-22 * "BRANTLY" <% 30>/;
-23 * "SKELVE" <% 30>/;
+14 * "JUMP" <% 30>/;
```

<% N> This tells DMDX to present the frame for N ticks. Actually, it tells DMDX when to schedule the next frame

If the tick duration was 16.67 (video card at 60Hz), 30 ticks would be 500ms (30 X 16.67 = 500)

If you do not include the <% N> parameter, the frame will be presented for the default frame duration

specified in the header (with the fd parameter)

Other ways to code Frame Duration

<ms% N>

this keyword specifies a frame duration as N in milliseconds. The result is always rounded *up* to the nearest whole number of ticks

This is very useful when you want to present a stimulus or a blank screen for 500 ms. You include for example : <ms% 500>. The following example of an item line presents a "+" for 250 ms, then a sound ("sing"), then waits for 500 ms, then presents another sound ("sang").

```
+212     <ms% 250> "+" / <wav 2> "sing" / <ms% 500> / <wav 2> "sang" * /;
```

<ctr% N>

this keyword specifies a frame duration using Counter number N's value as the number of ticks

Item lines [/]

```
+11 * “RABIES” <% 30>/;
-22 * “BRANTLY” <% 30>/;
-23 * “SKELVE” <% 30>/;
+14 * “JUMP” <% 30>/;
```

/ This marks the end of a frame. At this point the screen is cleared (i.e. after the word has been displayed for 500 ms)

If we omitted the /, then the word would stay on the screen until we next asked DMDX to display something else (in this case until the next line in the file).

! Can be used in a frame to indicate that the previous Frame should not be erased.

Item lines [;]

```
+11 * "RABIES" <% 30>/;
-22 * "BRANTLY" <% 30>/;
-23 * "SKELVE" <% 30>/;
+14 * "JUMP" <% 30>/;
```

; This marks the end of an item. At this point DMDX will wait for a response or a time-out and then move on to the next item.

Before displaying the next item DMDX waits for the delay time set in the header line (d parameter). [It uses this delay time to load the stimuli for the next item.]

In some of our examples, this was 120 ticks. Therefore:

$$120 \times 16.67 = 2000 \text{ milliseconds}$$

Item lines [Timing]

```
+11 * "RABIES" <% 30>/;
```

Two important notes with respect to item timing

1. **DMDX considers the item text that follows the / and precedes the ; to be a second frame. Nothing is presented in this frame and no <% N> keyword can be included b/c there is no additional frame after it to schedule. However, the frame will take one tick to execute.**

   You can consider this to be part of your ITI (inter-trial interval). Therefore, I always set my d parameter to be one tick less than I want the ITI to be. In this case, if I wanted a 2000 millisecond ITI, I would set d to 119 (rather than 120)

Item lines [Timing]

```
+11 * “RABIES” <% 30>/;
```

2. DMDX will wait for either a response or the timeout to expire (t parameter in header. Set to 1500 in our example) before starting the next item. If you do not want your overall ITI length to vary based on participants’ response speed, an additional consideration becomes important.

The timeout must be less than the time between the clockon (frame onset) and the end of the item.

See the modification to the sample item on the next page to accommodate this with a timeout of 1500…

<% 30> = 30 ticks (500 ms)

<% 60> = 60 ticks (1000 ms) so the overall duration between clock-on and end of item is about 1500 ms + 1 tick.

Item lines [0]

```
<ep> <cr> <fd 1> <d 59> <t 1500>  <id keyboard> <dbc
210210210> <dfs 36> <dwc 0> <vm 1024,768,768,16,60>
<nfb><rcot> <eop>

0 "Press SPACEBAR to start";
+11 * "RABIES" <% 30>/<% 60>/;
-22 * "BRANTLY" <% 30>/<% 60>/;
-23 * "SKELVE" <% 30>/<% 60>/;
+14 * "JUMP" <% 30>/<% 60>/;
0 "The END! Thank you for your participation";
```

Items that begin with 0 are used as message items and to pause the program. Even in continuous run mode, these items will pause until the participant (or experimenter) requests the next item by pressing spacebar (or other response mapped to request)

This can be over-ridden with the use of the <c> keyword

The complete item file: *Sample.rtf*

```
<NumberOfItems 8> <Scramble 4> <AZKII> <VideoMode 640,480,480,16,0> <ContinuousRun>
<Delay 36> <FrameDuration  150> <Timeout 2200> <id "keyboard"> <DefaultBackgroundColor
255255255> <DefaultWritingColor 0> <MapNegativeResponse "+Left Ctrl">
<MapPositiveResponse "+Right Ctrl">
$
0 <line -2> ”Instructions for visual lexical decision experiment”,<line -1>  “Press the
left Control Key on the Keyboard as quickly”,<line 0>  “as possible if you see a string of
letters on the screen that”,<line 1> “is not an existing word in English. Press the Right
Control Key”,<line 2> “if what you see is a real word”,<line 5> “Press SPACEBAR to start”;
$
+102001 * “ZOOLOGICAL” /;
-100102 * “WONKER” /;
+101003 * “BOTTLE” / ;
-100104 * “PIRAF” /;
$
0 ”Take a break...press SPACEBAR when ready”;
$
+101005 * “PUB” /;
-100106 * “FOLLESH” /;
+102007 * “PANDEMONIUM” / ;
-100108 * “DRAL” /;
$
0 <line -1> ”The End...thank you for participating”, <line 1> “Press Esc to Save the
data”;
$
```

Randomising the order of trials

```
<NumberOfItems 8> <Scramble 4> <AZKII> <VideoMode 640,480,480,16,0>
<ContinuousRun> <Delay 36> <FrameDuration  150> <Timeout 2200>
<id "keyboard"> <DefaultBackgroundColor 255255255> <DefaultWritingColor 0>

<MapNegativeResponse "+Left Ctrl"> <MapPositiveResponse "+Right Ctrl">
```

NumberOfItems

Scramble

These two parameters can be used to present items in a randomised order.

DMDX first divides the item file up into a number of blocks of items, the size of this block being determined by the parameters NumberOfItems and Scramble: if the NumberOfItems is 8 and the Scramble parameter is set to 4, then the number of blocks is 2. It then randomly orders the items within each block, and then finally randomly orders the blocks themselves.

The complete item file: messages for the subject

```
$
0 ”Take a break, press SPACEBAR when ready”;
$
```

To complete our experiment, we add some item lines to our experiment to serve as instruction displays. The $ before and after these lines tells DMDX not to include these items in the process of randomising the order of trials.

These item lines are set up the same way as other trials, but have item number 0. There’s no clock on signal, so DMDX will ignore left and right responses and wait until the spacebar is pressed.

The complete item file: long messages for the subject

```
$
0 <line -2> ”Instructions for visual lexical decision
experiment”,<line -1>  “Press the left Control Key on the
Keyboard as quickly”,<line 0>  “as possible if the string of
letters on the screen”,<line 1> “is not a word in English.
Press the Right Control Key”,<line 2> “if what you see is a
real word”,<line 5> “Press SPACEBAR to start”;
$
```

0 item lines may also include the instructions or other longer messages for the subject. line can be used to display multiple lines on the display at the same time. For example, <line 5> tells DMDX that this line should be displayed 5 lines below the center of the screen. Commas must be used to separate the different lines of text.

Note the general principle: multiple displays within a frame separated by a comma will be displayed simultaneously.

The complete item file: the end of the experiment

```
$
0 <line -1> ”The End...thank you for participating”,
<line 1> “Press Esc to Save the data”;
$
```

At the end of the experiment, DMDX will not move on when the spacebar is pressed. You will need to press the ESC-key on the keyboard. DMDX will then prompt you with a message box asking whether you want to save the data. If you press ‘yes’ the data will be saved in the same directory as the item file (*sample.rtf*) under the name *sample.azk*.

Note: if you press ESC during the experiment, DMDX will prompt you with a message asking if you wish to abort the experiment. It will save the data collected so far in a temporary file called *job1.zil*

Preventing the order of blocks being scrambled: \

```
<NumberOfItems 8> <Scramble 4> <AZKII> <VideoMode 640,480,480,16,0>
    <ContinuousRun> <Delay 36> <FrameDuration  150> <Timeout 2200> <id "keyboard">
    <DefaultBackgroundColor 255255255> <DefaultWritingColor 0> <MapNegativeResponse
    "+Left Ctrl"> <MapPositiveResponse "+Right Ctrl">
+102001 * “ZOOLOGICAL” /;
-100102 * “WONKER” /;
+101003 * “BOTTLE” / ;
-100104 * “PIRAF” /;
$
0 ”Take a break...press SPACEBAR when ready”;
$
\
+101005 * “PUB” /;
-100106 * “FOLLESH” /;
+102007 * “PANDEMONIUM” / ;
-100108 * “DRAL” /;
$
0 <line -1> ”The End...thank you for participating”, <line 1> “Press Esc to Save the
    data”;
$
```

More things you can do with DMDX

Item lines - other things you can do in a trial

Some examples:

- More than one stimulus
- Presenting Pictures
- Presenting Sounds

Item lines - more than one stimulus per trial, sequentially

```
+120101 "giraffe"/ * "ZOOLOGICAL" /;
-100202 "balloon"/ * "WONKER" /;
+110103 "wine"/ * "BOTTLE" / ;
-100204 "pizza"/* "PIRAF" /;
```

To display more than one frame per trial simply add another frame to each item line. In this case there would be a word displayed in lower case for one frame, followed by a larger word in upper case, for which a response will be measured (clock-on * included here).

Both frames would be displayed for the standard frame duration.

WYSIWYG: Note that the font and size etc. you use in your script file should be what will be displayed on the screen.

Item lines - more than one stimulus per trial, simultaneous

```
+300210 * <bmp> "H02", <Ln 1>   "heaven";
-410220 * <bmp> "S02", <Ln 1>   "exalted";
-310321 * <bmp> "S03", <Ln 1>   "hopeless";
```

Just put a comma between the items within a frame (as in the emotion words experiment)

*Note also: change format of text **only within the quotes**.*

This won't work because the first quote sign of the word is formatted in blue too:

```
-310321 * <bmp> "S03", <Ln 1>   "hopeless";
```

Item lines: non default display durations, positions

```
+101001 * <ms% 1000> “BOTTLE” /;
+102002 * <ms% 500> “ZOOLOGICAL” /;
-100103 * <Line -1>“WONKER” /;
-100104 * <Line 2> “PIRAF” /;
```

`<ms% 1000>` This specifies the duration of the frame in ms
(You can do it in ticks instead)

`<Line -1>` Centered 1 line above the centre of the screen

`<Line 2>` Centered 2 lines below the centre of the screen

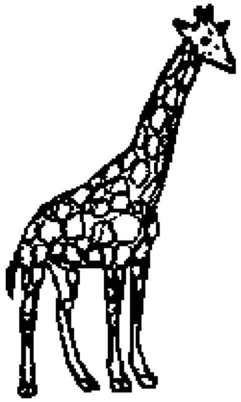
Item lines: blank intervals

```
+101001 <ms% 250> "nurse" / <ms% 250> / "DOCTOR";
```

/ <ms% 250> / introduces a blank frame interval of 250 ms between the two word displays

Item lines - presenting pictures

```
+100010001 * <bmp> "giraffe" ;
-100020002 * <bmp> ”bottle" ;
```



<bmp> This keyword tells DMDX to display a picture on the screen for the default frame duration

Picture files are stored as **.bmp** files. So to run this bit of the experiment we would need the following files to be in the directory:

giraffe.bmp, bottle.bmp

Item lines - picture position and size

+100010001 * **<bmp 0.25, 0.25>** "giraffe" ;

-100020002 * **<bmp 160, 120>** ”bottle" ;

<bmp 0.25, 0.25> displays the picture a quarter from the top and left of the screen (top left corner)

<bmp 160, 120> ditto, but specified in pixels, so you have to work in out from the screen resolution in the header.

Item lines - presenting sounds

+10010001 * **<wav 2>** ”zoological" /;

-10020002 * **<wav 2>** ”shoulteer" /;

<wav 2> This keyword tells DMDX to play out a speech file.

Speech files are stored as **.wav** files. So to run this bit of script we would need the following files in the same directory:

zoological.wav, wonker.wav

+100801 * <wav 2> <svp start> "H01"/ <bmp> "H08";

Item lines: synchronising sound onset and visual display

- Exercises:
  - Create a *lexical decision* experiment with
    - *Four words and four non-words per block*
    - *A coding system reflecting word/non-word status and frequency.*
    - *Five blocks*
    - *Beginning, end, and block instructions*
    - *Randomize the trials within blocks but not the blocks*
  - Create a *subliminal priming* experiment with
    - *40ms duration prime followed by a response-terminated target.*
    - *Lexical decision of semantically related and unrelated targets.*
    - *Suitable coding system*
    - *Six blocks of ten items.*
  - Create a "living vs. non-living" picture categorisation task with a distracting auditory secondary task. (See C:\WINDOWS\MEDIA\ for files).

- Exercises: Submission of coursework.
  - Exercises are assessed. Please upload your .rtf files and any associated .bmp and .wav files on the Oncourse site
  - The “official” deadline for these things is in two weeks. However, if you can’t get them to work, you should bring it up at the next class.

- Final note
  - .bmp is one of the most common format for pictures.
  - .wav is – obviously – very common for Laboratory Phonology

Item lines - visual/visual priming

```
+101 "wobble" / * "RABIES" /;
+201 "cove" / * "BRANTLY" /;
+202 "char" / * "SKELVE" /;
+1 "jumped" / * "JUMP" /;
```

To display more than one word at a time simply add another frame to each item line. So now there will be a word displayed in lower case for one frame, followed by another word in upper case, for which a response will be measured (clock-on signal * included here).

Item lines - masked priming

```
+101 "#########" / <fd 4> "wobble" / * "RABIES" /;
+201 "#########" / <fd 4> "cove" / * "BRANTLY" /;
+202 "#########" / <fd 4> "char" / * "SKELVE" /;
+1 "#########" / <fd 4> "jumped" / * "JUMP" /;
```

"#########" / This frame contains a pattern mask (actually just a row of # marks) to be displayed before the prime word as used in masked priming.

Adjust frame durations (here <fd 4> is about 66 ms for the prime word) for your masked priming experiment.

Item lines - masked priming

```
+101 "#########" / <fd 4> "wobble" / * "RABIES" /;
+201 "#########" / <fd 4> "cove" / * "BRANTLY" /;
+202 "#########" / <fd 4> "char" / * "SKELVE" /;
+1 "#########" / <fd 4> "jumped" / * "JUMP" /;
```

<fd 4> This keyword tells DMDX to display this frame for only 4 ticks (approx. 47milliseconds)

When combined with the preceding mask the prime word will be imperceptible to subjects.

Item lines - cross-modal priming

```
+1 <wav 2> "camel" / * "LOBSTER" /;
-2 <wav 2> "truck" / * "TRUNT" /;
+3 <wav 2> "blackberry" / * "BLACKBERRY" /;
-4 <wav 2> "carrot" / * "PRONK" /;
```

<wav 2> This keyword tells DMDX to play out a speech file.

Now DMDX will play a speech file, and then display a word at the **end** of the speech file.

Speech files are stored as **.wav** files. So to run this file we need the following files to be in the directory:

camel.wav, truck.wav, blackberry.wav, carrot.wav

Item lines - cross-modal priming

```
+1 <wav 2> <svp start> "camel" / * "LOBSTER" /;
-2 <wav 2> <svp start> "truck" / * "TRUNT" /;
+3 <wav 2> <svp start> "blackberry" / * "BLACKBERRY" /;
-4 <wav 2> <svp start> "carrot" / * "PRONK" /;
```

<svp start> This keyword tells DMDX to move on to the next frame at the same time as playing a speech file.

Now DMDX will play a speech file, and display a word at the **beginning** of the speech file.

picture.rtf

Item lines - picture priming

```
-1 <bmp> "hippo" / * "LOBSTER" /;
+2 <bmp> "truck" / * "TRUCK" /;
+3 <bmp> "blackberry" / * "BLACKBERRY" /;
-4 <bmp> "carrot" / * "JUMP" /;
```

<bmp> This keyword tells DMDX to display a picture on the screen for the default frame duration

Picture files are stored as **.bmp** files. So to run this experiment we will need the following files to be in the directory:

hippo.bmp, truck.bmp, blackberry.bmp, carrot.bmp