FiveChoice

A Whisker client

by Rudolf Cardinal

www.whiskercontrol.com

Distributed by Campden Instruments Ltd (www.campden-inst.com)

FiveChoice

Printed: May 2017 in Cambridge, UK

Creator (Whisker)

Rudolf N. Cardinal

Design and Programming (Whisker)

Rudolf N. Cardinal

Michael R. F. Aitken

Legal Advisor (CUTS)

Adjoa D. Tamakloe

Sales (Campden)

Julie Gill

Contacting the authors:

For information about Whisker, visit http://www.whiskercontrol.com/.

If you have sales enquiries about Whisker, contact Campden Instruments Ltd at http://www.campden-inst.com/.

If you have comments or technical enquiries that cannot be answered by the sales team, contact the authors:

Rudolf Cardinal (rudolf@pobox.com)
Mike Aitken (m.aitken@psychol.cam.ac.uk)

Table of Contents

Foreword

WARNING

Whisker is a system designed for research purposes only, and should never be used to control medical apparatus or other devices that could endanger human life.

DISCLAIMER

The authors, copyright holders, and distributors disclaim all responsibility for any adverse effects that may occur as a result of a user disregarding the above warning.

1 FiveChoice

1.1 About FiveChoice

Purpose

Five-choice serial reaction time task.

Citing FiveChoice

Please cite as, for example:

(in text)
The task was implemented in the FiveChoice program (version 5.3, R.N. Cardinal) using the Whisker control system (Cardinal & Aitken, 2010).

(in bibliography)
Cardinal RN, Aitken MRF (2010). Whisker: a client–server high-performance multimedia research control system. *Behavior Research Methods* 42: 1059–1071. (PubMed ID: 21139173. DOI:10.3758/BRM.42.4.1059.)

Software requirements

Requires Whisker v2.0 or greater.

Data storage

- Text-based output to disk.
- ODBC data storage to a database (supplied).

Author

Rudolf Cardinal (rudolf@pobox.com).

Copyright

Version history

- v1.1 - First version finished 25 Mar 2002
- v2.4 - First version actually used! (26 Mar 2004)
- v2.5 - minor bugfix to stop program complaining about wrong XML config versions; bug fix to ensure that timer events have phase checks (stops a bug in which a trial could be classified as being both correct and an omission); added "preferred box" field; more progress info on main dialogue
- v2.6 (8 June 2004): option to punish perseverative responses following correct responses; minor user interface changes
- v2.7 (10 June 2004): option to debounce incoming inputs
- v2.8 (2 Sep 2004): yoking (and more things are recorded with the standard task).
- v2.9 (14 Feb 2005): forced-choice task added; pseudorandom location selection added.
- v3.0 (10 Oct 2005): option not to record line OFF events.
- v3.1 (14 Mar 2006): Bug fixed: wasn't recording events from yoked boxes numbered 10 or above.
- v3.2 (4-6 Apr 2006): Help improved; options for variable stimulus duration within a session; more draw-without-replacement pseudorandom options added.
- v3.3 (7 Apr 2006): Improved option for true zero stimulus duration; running display of perseverative

responses.

- v3.4 (18 May 2006): Better control of session end time.
- v3.5 (25 May 2006): Front panel during pre-stimulus/post-stimulus timeout is now only *optionally* considered premature/perseverative, respectively.
- v3.6 (4 July 2006): Bug fixed: was possible to set up a yoked box having the same box number as the master box, or as another yoked box.
- v3.7 (8 March 2007): Easier compilation by user.
- v3.8 (6 July 2007): Minor pellet-counting cosmetic bug fixed.
- v4.0 (10-13 June 2008): computer-controlled dimmers for lights/noise distractors implemented.
- v5.0 (12 Jan 2009): Server default changed from "loopback" to "localhost" (Windows Vista compatibility and more general standardization).
- v5.1 (30 Apr 2009): Strobe hardware build target. **See version tracker.**
- v5.2 (28 Feb 2010): Updated to compile under VS2008. Also database fields `FiveChoice_Config.PelletPulseLength_ms` and `FiveChoice_Config.InterpelletGap_ms` changed from single (float) to long (int).
- v5.3 (2 May 2011): Now allows exact specification of holes to use for stimulus presentation (i.e. from 1-choice to 5-choice). Database changes to match (see version tracker).
- v5.4 (19 Oct 2011). "PerseverativeNosepokes," missing from CSV header output in summary file - data was there, but header was missing (so subsequently mis-aligned) - fixed. (Database was unaffected.) See also accompanying Python rescue script. // Date/time string stored in textfile now includes seconds fields.

1.2 Required devices

The program requires to claim devices in groups named **box0, box1, box2...** with device names as listed below in bold:

```
# ---------------- Box 0 definition
# INPUTS
line    0       box0    REARPANEL
line    3       box0    HOLE_0
line    6       box0    HOLE_1
line    9       box0    HOLE_2
line    12      box0    HOLE_3
line    15      box0    HOLE_4

# OUTPUTS
line    24      box0    HOUSELIGHT
line    27      box0    PELLET
line    30      box0    TRAYLIGHT
line    33      box0    TONE
line    36      box0    STIMLIGHT_0
line    39      box0    STIMLIGHT_1
line    42      box0    STIMLIGHT_2
line    45      box0    STIMLIGHT_3
line    48      box0    STIMLIGHT_4
line    51      box0    STIMLIGHT_0_DIM
line    54      box0    STIMLIGHT_1_DIM
line    57      box0    STIMLIGHT_2_DIM
line    60      box0    STIMLIGHT_3_DIM
line    63      box0    STIMLIGHT_4_DIM
line    66      box0    DIM0
line    69      box0    DIM1
line    72      box0    DIM2

# ... and so on for all your boxes
```

Please ensure that these devices are available and listed in the device definition file in use by the server. (The snippet above shows an extract from a typical definition file.)
The **TONE** device is the one used as a tone or a white-noise distractor.

MORE DETAIL

Please note that for **nine-hole boxes,** the five-choice task is usually configured to use holes 1, 3, 5, 7, and 9. These boxes often also have a **dimming system**. The three types of box in common use are as follows:

Box type 1: no dimming system

- In the WhiskerServer device definition file, assign your chosen five holes so that their input lines (nosepoke detectors) map to HOLE_0 to HOLE_4.
- Assign their corresponding output lines (lamps) to STIMLIGHT_0 through STIMLIGHT_4.
- Assign all the other devices (REARPANEL, HOUSELIGHT, PELLET, TRAYLIGHT, TONE).
- You will need to create **fake lines** in WhiskerServer, and assign STIMLIGHT_0_DIM to STIMLIGHT_4_DIM to them.
- Likewise, you will need to assign DIM0, DIM1, and DIM2 to fake lines.

Box type 2: nine-hole box with "old" dimming system (e.g. Cambridge Cognition / Campden Instruments 80600 [rat] and 80610 [mouse] Nine Hole Box Operant Chamber)

- In nine-hole boxes, holes 1, 3, 5, 7, and 9 are usually used for the five-choice task. So:
- Assign your box's input HOLE1 to have the label HOLE_0.
- Assign your box's input HOLE3 to have the label HOLE_1.
- Assign your box's input HOLE5 to have the label HOLE_2.
- Assign your box's input HOLE7 to have the label HOLE_3.
- Assign your box's input HOLE9 to have the label HOLE_4.
- Assign your box's output LAMP1A to have the label STIMLIGHT_0.
- Assign your box's output LAMP3A to have the label STIMLIGHT_1.
- Assign your box's output LAMP5A to have the label STIMLIGHT_2.
- Assign your box's output LAMP7A to have the label STIMLIGHT_3.
- Assign your box's output LAMP9A to have the label STIMLIGHT_4.
- Assign your box's output LAMP1B to have the label STIMLIGHT_0_DIM.
- Assign your box's output LAMP3B to have the label STIMLIGHT_1_DIM.
- Assign your box's output LAMP5B to have the label STIMLIGHT_2_DIM.
- Assign your box's output LAMP7B to have the label STIMLIGHT_3_DIM.
- Assign your box's output LAMP9B to have the label STIMLIGHT_4_DIM.
- Assign all the other devices (REARPANEL, HOUSELIGHT, PELLET, TRAYLIGHT, TONE).
- You will need to create **fake lines** in WhiskerServer, and assign DIM0, DIM1, and DIM2 to them.
- Adjust your box's DIAL A or V1 to set what will be known as "brightness level 1".
- Adjust your box's DIAL B or V2 to set what will be known as "brightness level 2".

Box type 3: nine-hole box with "new" dimming system (e.g. Cambridge Cognition / Campden Instruments 80600A [rat] and 80610A [mouse] Nine Hole Box Operant Chamber)

- In nine-hole boxes, holes 1, 3, 5, 7, and 9 are usually used for the five-choice task. So:
- Assign your box's input HOLE1 to have the label HOLE_0.
- Assign your box's input HOLE3 to have the label HOLE_1.
- Assign your box's input HOLE5 to have the label HOLE_2.
- Assign your box's input HOLE7 to have the label HOLE_3.
- Assign your box's input HOLE9 to have the label HOLE_4.
- Assign your box's output LAMP1 to have the label STIMLIGHT_0.

- Assign your box's output LAMP3 to have the label STIMLIGHT_1.
- Assign your box's output LAMP5 to have the label STIMLIGHT_2.
- Assign your box's output LAMP7 to have the label STIMLIGHT_3.
- Assign your box's output LAMP9 to have the label STIMLIGHT_4.
- Assign DIM0, DIM1, and DIM2.
- Assign all the other devices (REARPANEL, HOUSELIGHT, PELLET, TRAYLIGHT, TONE).
- You will need to create **fake lines** in WhiskerServer, and assign STIMLIGHT_0_DIM through STIMLIGHT_4_DIM to them.

1.3 Dimming system

A variety of systems for dimming the stimulus lights (and sometimes the tone/noise generator) exist.

Box type 1: no dimming system

- No control is possible over the intensity of the stimulus lights; they're either on or off.
- No control is possible over the intensity of the tone/noise; it's either on or off.

Box type 2: nine-hole box with "old" dimming system (e.g. Cambridge Cognition / Campden Instruments 80600 [rat] and 80610 [mouse] Nine Hole Box Operant Chamber)

- Each of the holes has an 'A' line and a 'B' line controlling its lamp (so, hole 1 has lines known as LAMP1A and LAMP1B, for example).
- There are two manual dials in the box (not under computer control), labelled DIAL A (or V1) and DIAL B (or V2).
- For a given hole, if the A and B lines are both off, then the stimulus light is off.
- For a given hole, if the A line is on and the B line is off, then the stimulus light is on at full power. *The FiveChoice task refers to this as dimming level 0.*
- For a given hole, if the A and B lines are both on, then the stimulus light is on with its intensity as set by the V1/A dial.*The FiveChoice task refers to this as dimming level 1.*
- For a given hole, if the A line is off and the B line is on, then the stimulus light is on with its intensity as set by the V2/B dial.*The FiveChoice task refers to this as dimming level 2.*
- No computer control is possible over the intensity of the tone/noise; from the computer's point of view, it's either on or off. Noise intensity may be controlled manually by setting it within the box.

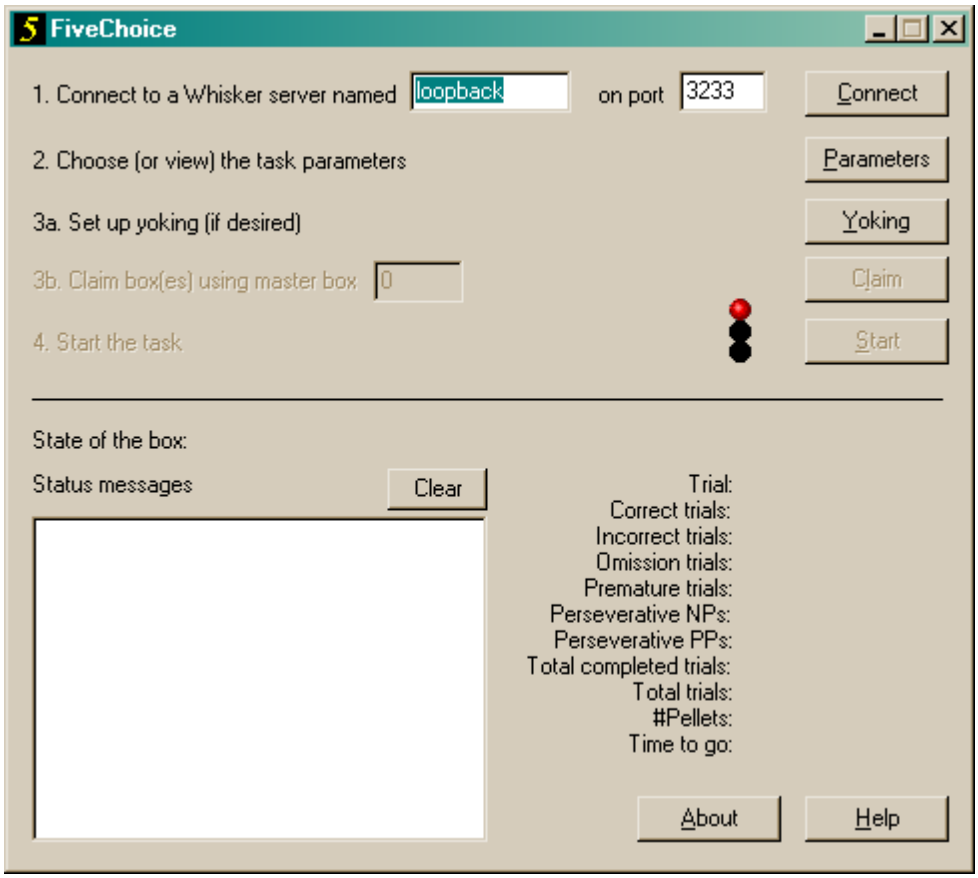
Box type 3: nine-hole box with "new" dimming system (e.g. Cambridge Cognition / Campden Instruments 80600A [rat] and 80610A [mouse] Nine Hole Box Operant Chamber)

- In this box, there is one control line for each stimulus light and one for the tone/noise generator.
- A variety of intensity control systems are possible, but the factory-set one (which is the most flexible, and which the FiveChoice task assumes) allows 8 levels of intensity control from full power through to minimum power, plus off.
- Three lines (DIM2, DIM1, DIM0) control the intensity for all devices. DIM2 is the most-significant-bit, and DIM0 is the least-significant bit (so, for example, if DIM2 was on and the others were off, we would write this in binary as 100 and call it the decimal number 4).
- The DIM lines can be set to 0 (full power) through to 7 (maximum attenuation, but still on). When a device (lamp or tone/noise generator) is turned on, it comes on with an intensity as specified by the DIM lines.
- For lights, intensities 0-7 represent 100%, 90%, 80%, 70%, 60%, 50%, 40%, 30% respectively (where 100% approximates a 24V, 3W bulb).
- For the tone/noise generator, intensities 0-7 represent 105dB, 100dB, 95dB, 90dB, 85dB, 80dB, 75dB, 70dB (the Campden documentation is vague as to the meaning of "dB"; perhaps dB SPL?).
- Once a device is switched on, it is recommended to leave the DIM lines unmodified for **5 ms** (as

of June 2008 according to Campden; the FiveChoice task supports this as a configurable parameter, but [as of 13 June 2008] defaults to **10 ms** for an extra safety margin).

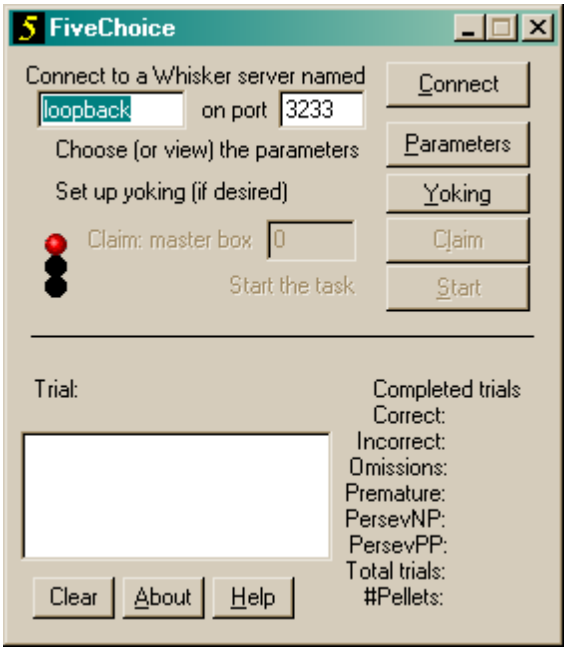
1.4 Using the task

When you run the task, the main screen looks as follows:



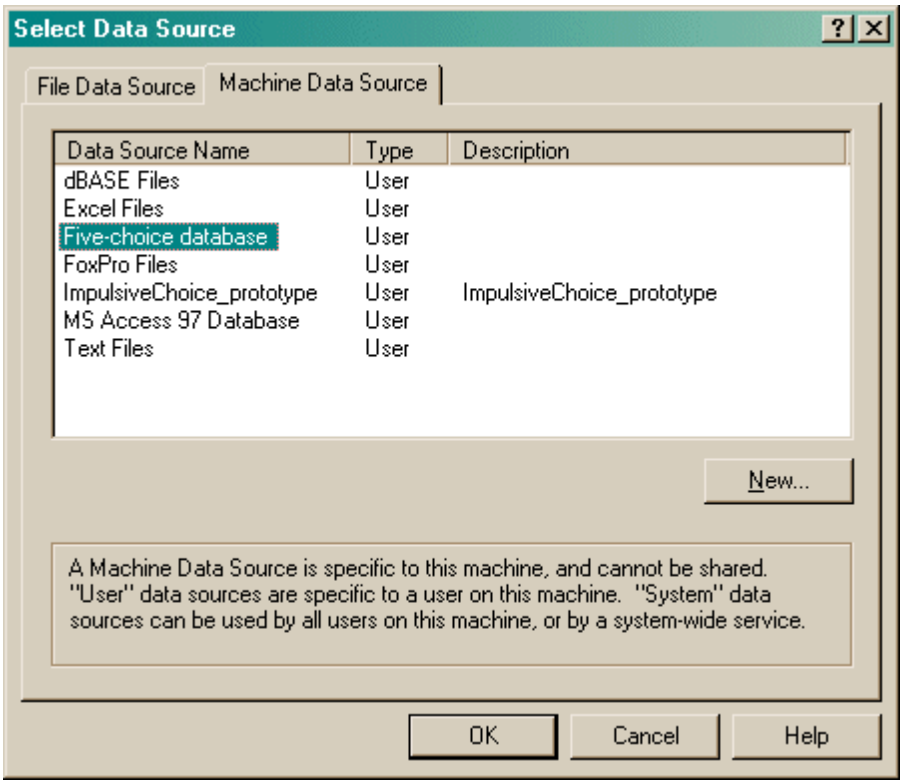
"NPs" means nosepokes (responses at the front holes); "PPs" means panel pushes (responses at the food alcove); see the Task Design.

There is an alternative version of the task (called **FiveChoice_SmallDialogue.exe,** rather than **FiveChoice.exe**) which takes up less space on the screen:



You must connect to a Whisker server, claim an operant chamber (box), and set up the parameters for your task. Optionally, you may set up yoking. Once that's done, the traffic lights will turn amber. When you are ready, press *Start* to begin the task.

When the task finishes, it saves data to disk and pops up a new dialogue box for you to select a database to store the data to. (The data sources are configured under *Control Panel* → *ODBC*.) If you previously specified an ODBC data source in the parameters, that data source is used automatically and you will only see a dialogue box if something goes wrong and the program needs your input.



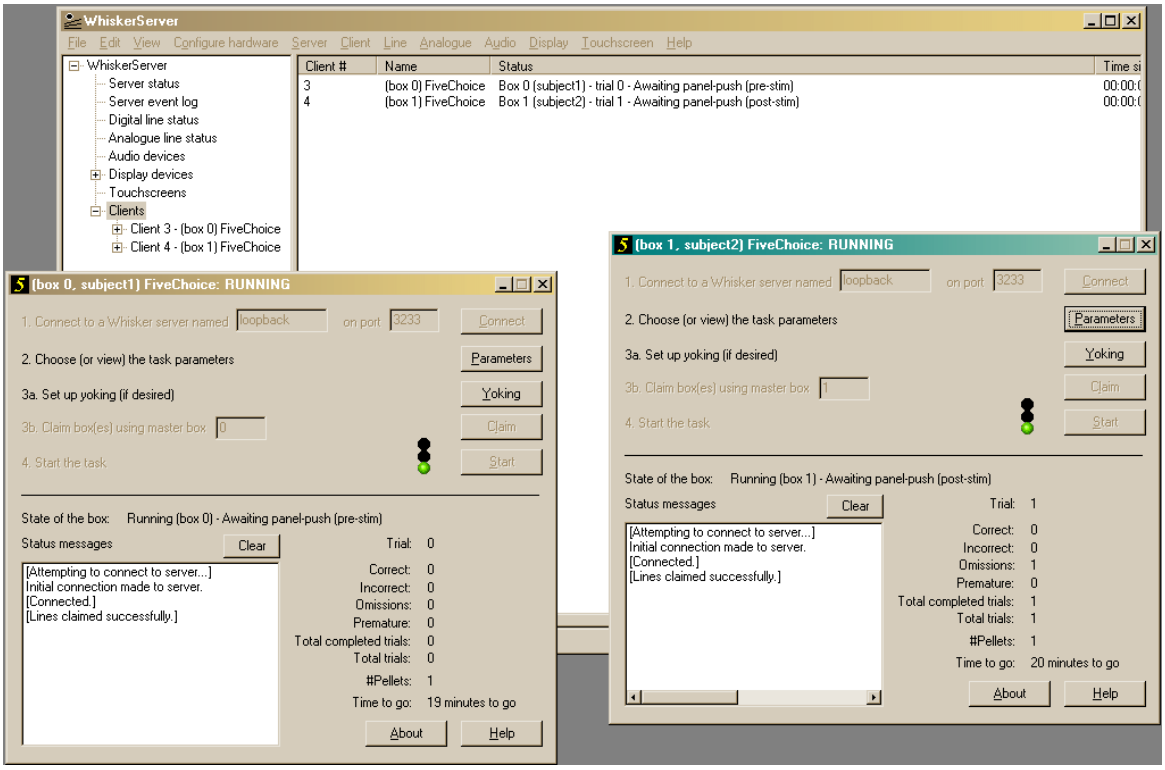
1.5 Running multiple boxes

The FiveChoice program controls **one** box in which the subject performs the full five-choice task. The same program can also control **multiple yoked** boxes, in which the "yoked" subject(s) experience the same events as their single "master" subject, but the yoked subjects do not influence the events.

To run the task with **multiple boxes**, meaning **multiple five-choice tasks**, or **multiple master/ yoked groups**, you simply need to run **multiple copies of the FiveChoice program**.

For example, suppose you have 6 five-choice chambers (boxes), and you have started the Whisker server. You could run copy 1 of the FiveChoice program, and have it claim box 1, load subject 1's configuration, and start it. You could then run a second copy of the FiveChoice program at the same time, claiming box 2, loading subject 2's configuration, and starting it... and so on.

An example is shown below, in which two subjects are running ("subject1" in box 0, and "subject2" in box 1). WhiskerServer is shown in the background



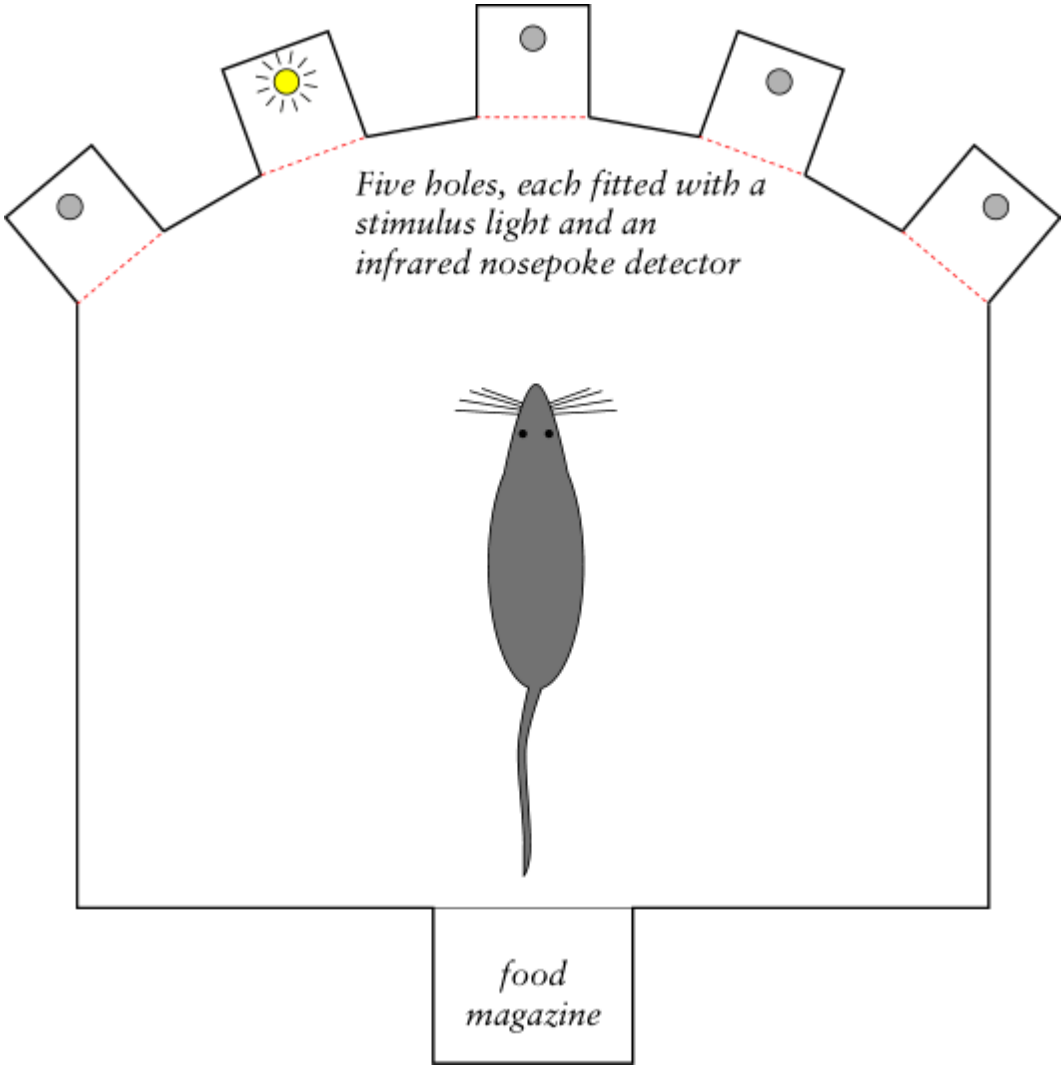
1.6 Task design and trial overview

History

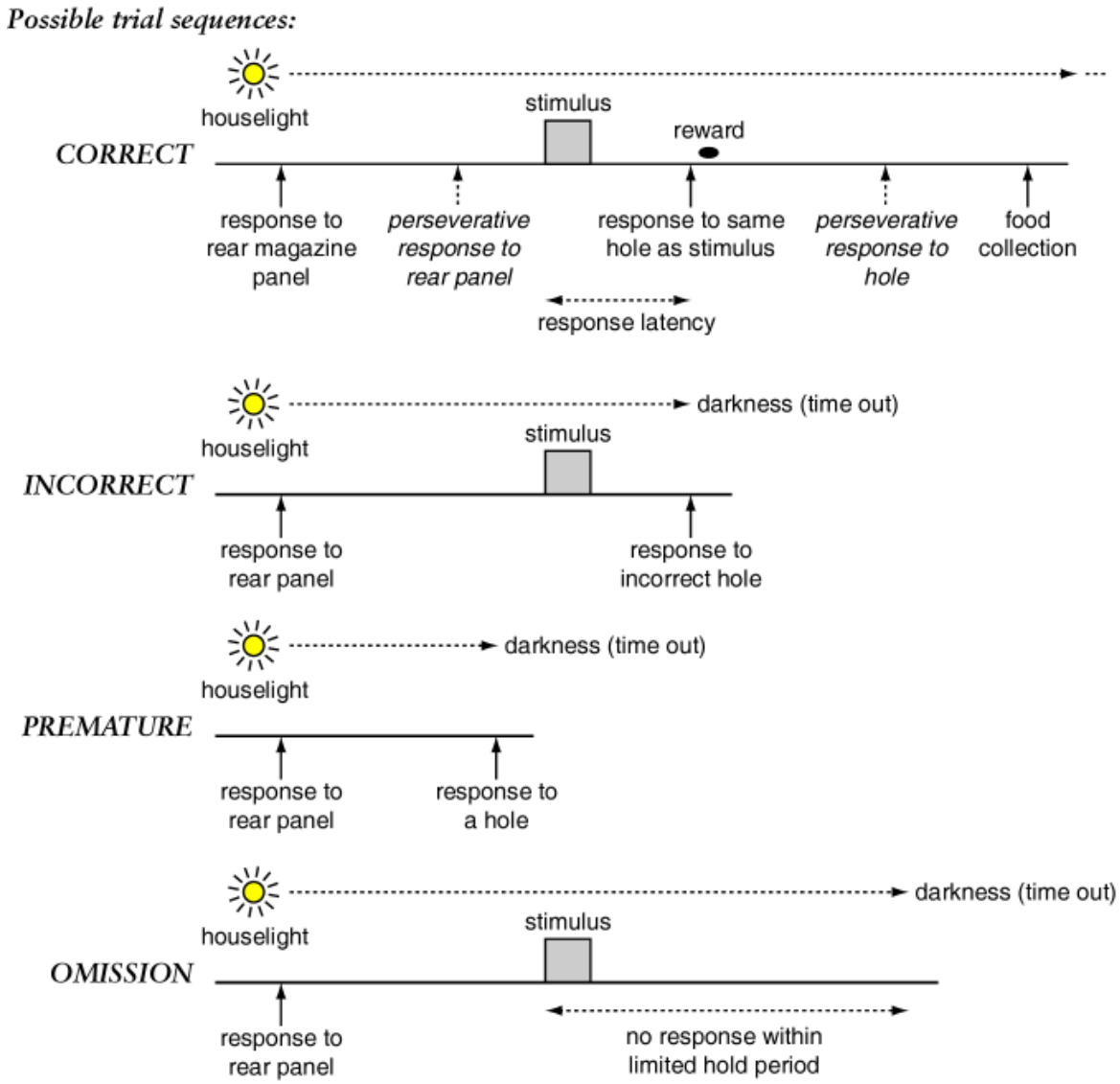
- Original rat version: Carli M, Robbins TW, Evenden JL, Everitt BJ (1983). Effects of lesions to ascending noradrenergic neurones on performance of a five-choice serial reaction time task in rats: implications for theories of dorsal noradrenergic bundle function based on selective attention and arousal. *Behav Brain Res* **9:** 361-380.
- Analagous to human continuous performance tests: Rosvold HE, Mirsky AF, Sarason I, Bransome EB, Beck LH (1956) A continuous performance test of brain damage. *J Consult Psychol* **20:** 343-350.

- First human five-choice task was by Leonard (see e.g. Wilkinson RT [1963] Interaction of noise with knowledge of results and sleep deprivation. *J Exp Psychol* **66:** 332-337).
- Review: Robbins TW (2002). The 5-choice serial reaction time task: behavioural pharmacology and functional neurochemistry. *Psychopharmacology* **163:** 362-380.

Apparatus



Task outline



See the trial details for a full description.

1.7 Trial details

Trial format

Think of the task as existing in a number of *states*, because that's how the program implements it. Incidentally, this is an exercise in mental discipline: if you can't specify a state table like this, having thought about what should happen in response to all possible events in any state, then you can't program the task properly.

Blue text shows the states that a successful subject cycles through, once it's completed the first trial progression of PRESTIM_PLEASEPUSH → INITIAL_PAUSE.

State name	Description	State of the box	Consequence of nosepoking at the front panel (one	Consequence of nosepoking at the rear (food)	Consequence of time passing

			of the five holes)	**panel**	
NOTSTARTED	Not started yet. Awaiting experimenter to start task.	Darkness.	-	-	-
PRESTIM_PLEASEPUSH	**Task begins here.** (For the first trial, a free pellet is given and the trial number is initially set to 0.) Rat must respond to rear panel to initiate the trial.	Houselight on. Traylight on (if used).	Score **premature response.** Perhaps punish by going to PRESTIM_TIMEOUT, which will eventually start a new trial. **(2)**	Go to INITIAL_PAUSE.	-
POSTSTIM_PLEASEPUSH	Post-stimulus punishment timeout over. Rat must respond to rear panel to initiate new trial.	Houselight on. Traylight on (if used).	Score **perseverative response.** Perhaps punish by going to POSTSTIM_TIMEOUT, which will eventually start a new trial. **(2)**	Go to INITIAL_PAUSE.	-
INITIAL_PAUSE	**Trials begin here. Trial number incremented.** Variable time before stimulus is presented (sometimes called the "ITI").	Houselight on.	Score **premature response** and enter PRESTIM_TIMEOUT, which will eventually start a new trial.	Score **perseverative panel-push.** No other consequence.	Go to STIM_ON.
STIM_ON	Stimulus presented briefly. Opportunity to respond.	Houselight on. One target hole lit.	Either a **correct** or an **incorrect** response. If correct, reward and go to AWAITING_COLLECT. If incorrect, go to POSTSTIM_TIMEOUT, which will eventually start a new trial.	Score **perseverative panel-push.** No other consequence.	Go to STIM_OFF.
STIM_OFF	Stimulus has gone off, but opportunity to respond remains.	Houselight on.	Exactly as for STIM_ON.	Score **perseverative panel-push.** No other consequence.	Score an **omission** and go to POSTSTIM_TIMEOUT, which will eventually start a new trial.
AWAITING_COLLECT	Successful response; subject has been rewarded. Awaiting food collection (which will also initiate the next trial).	Houselight on. Traylight on (if used).	Score **perseverative response.** Either no other consequence, or punish by going to POSTSTIM_TIMEOUT. **(3)**	Successful reward collection. **Begin a new trial** by entering INITIAL_PAUSE state.	-

PRESTIM_TIMEOUT	Rat is being punished for a premature nosepoke to the front panel.	Darkness.	Perhaps score **premature response (4)**, and perhaps restart PRESTIM_TIMEOUT **(1)**.	Recorded. No other consequence.	Go to PRESTIM_PLEASEPUSH.
POSTSTIM_TIMEOUT	Rat is being punished for an incorrect response, or a perseverative response.	Darkness.	Perhaps score **perseverative response (5)**, and perhaps restart POSTSTIM_TIMEOUT **(1)**.	Recorded. No other consequence.	Go to POSTSTIM_PLEASEPUSH.
FINISHED	Rat finished the task	Darkness.	-	-	-
ABORTED	User aborted the task	Darkness.	-	-	-

(1) Option: front panel responses during timeouts can prolong the timeout by restarting it (default) or not. On the Parameters dialogue, this is the option labelled "Front panel responding during timeouts prolongs (restarts) the timeout."
(2) Option: front panel responses while waiting to start a trial are punished with a timeout, or not (default). On the Parameters dialogue, this is the option labelled "Front panel responding while waiting to start trial is punished."
(3) Option: "Punish perseverative nosepokes following a correct response." See the Parameters dialogue.
(4) Option: "Front panel responding in a pre-stimulus timeout is scored as premature." See the Parameters dialogue.
(5) Option: "Front panel responding in a post-stimulus timeout is scored as perseverative." See the Parameters dialogue.

The duration of the STIM_ON and STIM_OFF times, together, constitute a **"limited hold"** period for the rat to respond.

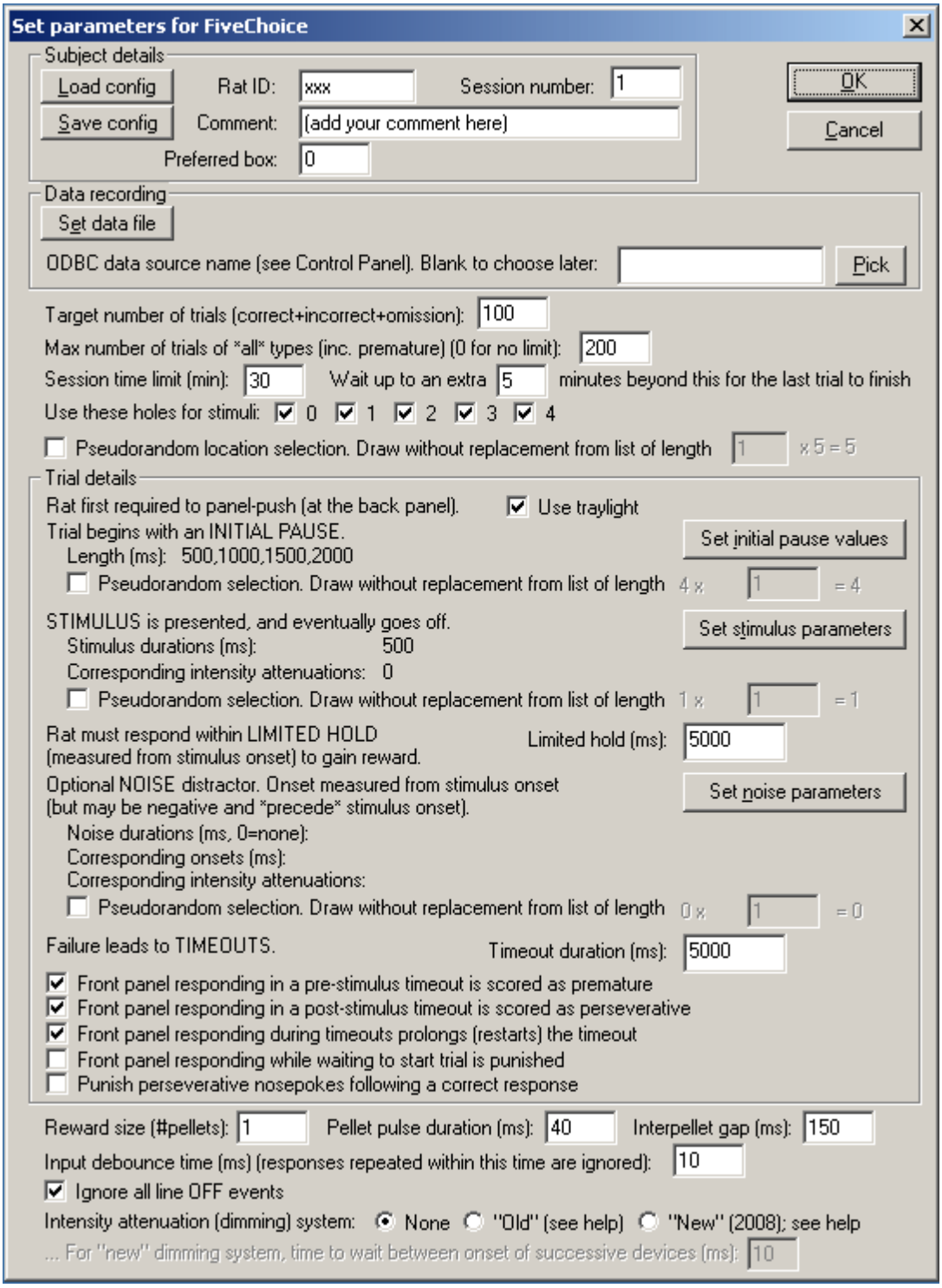
The task records every response with its location and the state (phase) the box was in when the response was made.

Additionally,

- there can be an overall time limit for the session;
- there can be a trial limit for the session;
- a noise (e.g. tone or white noise, depending on your device) can be presented as a distractor at various times (measured relative to stimulus onset and specified in the task parameters).

1.8 Parameters

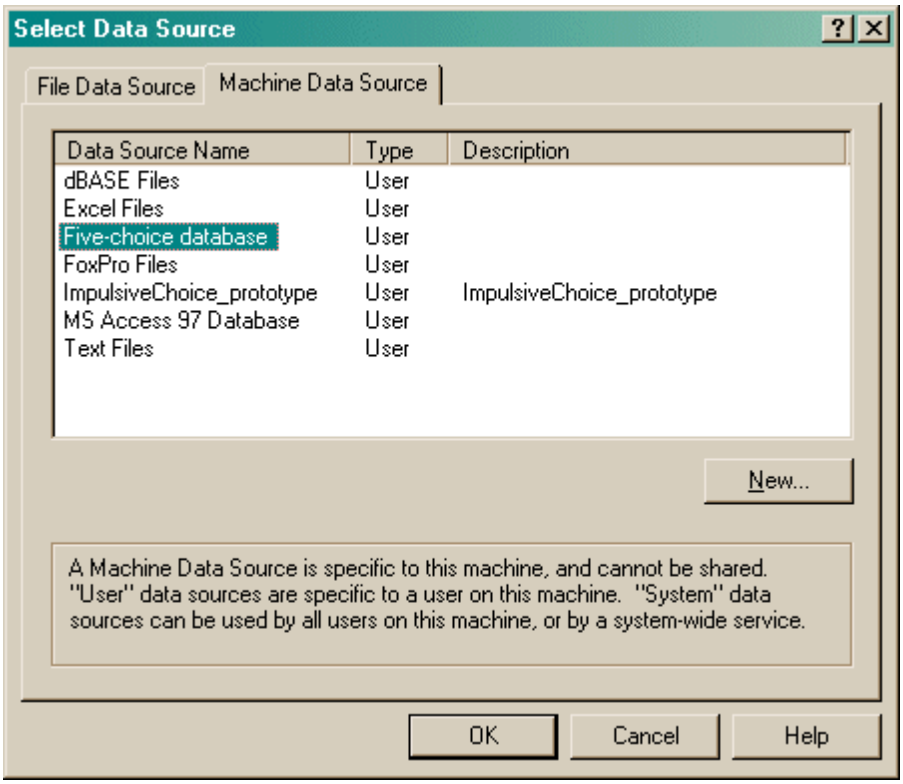
The parameters dialogue box looks like this:



- **SUBJECT DETAILS.** Here, you can enter your subject's ID and a comment or description, choose the default box (operant chamber) number that this subject usually runs in, and set its session number. You can also **load** and **save** the whole configuration (with the subject's details and all the information you can see on this screen). We recommend that you set up **one configuration file per subject.** If you do so, then you would set up the subject's task parameters once, then save

the configuration file. Every time you reload the configuration, on subsequent sessions, the subject's parameters will be correctly recalled and the program will automatically increase the session number by one. Unless you want to change the task parameters, this makes re-running a subject a very rapid procedure.

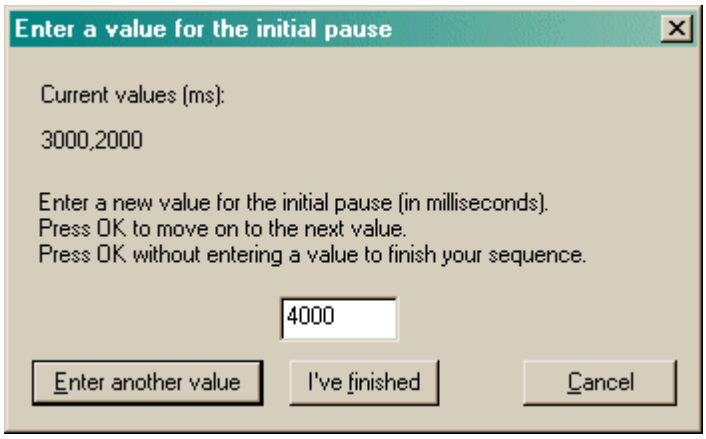
- **DATA RECORDING.** The FiveChoice program stores its results in two places: a text file, and a relational database. Here, you can specify the **filename** of the text file. Note also that a unique filename for the text file is generated whenever you load a subject's configuration, so you shouldn't routinely need to enter this. You can also specify the **ODBC** name of the database (see the Results section of this help for more information). Click "Pick" to choose from a list of ODBC databases configured on your computer (see below). Your choice will be recorded and will apply to this subject from now on, or until you specify a different database. If you don't specify a database now, or you delete the value in the "ODBC data source name" box, you'll be asked to choose a database when the task ends (and that choice will only apply to the session in progress).



- **Target number of trials.** Set the maximum number of correct/incorrect/omission trials. The program will terminate when this target has been reached.
- **Maximum number of trials of all types.** You can set a further optional limit here of the maximum number of trials of *all* types, including "premature" trials (see Task description and Trial details). Enter 0 for no such limit.
- **Session time limit.** Set the maximum session time. The program will terminate when this time limit is reached, after finishing any trial currently in progress.
- **Session extra time.** After the session time expires, the program will normally wait (in "extra time") for the current trial to finish. You can specify how much "extra time" to allow for this here. After the "extra time" expires, the program will terminate immediately, interrupting any trial that is still in progress.
- **Use these holes for stimuli...** This option allows you to choose the holes in which targets will be presented (i.e. from a 1-choice to a 5-choice task). Responding in other holes is still noticed by the task and acted upon.
- **Pseudorandom location selection.** By default, the location is chosen truly at random for each trial. However, if you tick "Pseudorandom...", then the program behaves as follows. It makes a list containing N copies of each of the possible target locations, where N is the number you can type

into the box (labelled "... from [a] list of length *y* x **N**", where *y* is the number of holes in use). For each trial, the target location is drawn at random, without replacement ("draw without replacement"), from this list. When the list is emptied, it is repopulated with *y*N entries as before. The idea behind this is to allow you to prevent the program from choosing the location completely at random, instead ensuring an exactly equal distribution of locations across a set of trials, nevertheless with some random variation from trial to trial. (Obviously, if you specify N=1, then since you have *y* possible locations, each possible location will be used once in every *y* trials.) The larger N is, the closer the system is to true random sampling (i.e. an infinite N is equivalent to unticking this option). Be aware, however, that true random choice (choice independent for each trial) means that your subject cannot predict anything on the basis of past locations (even if the distribution of locations across trials is not exactly even as a consequence of random sampling), but with a draw-without-replacement system, location *does* become informative. For a more extensive description of this technique, see Randomness, pseudorandomness, and drawing without replacement.

- **TRIAL DETAILS.** For full details of the task, see the Task description and Trial details.
- The task begins by requiring the subject to respond at the rear panel (the food alcove). Tick **use traylight** if you wish the traylight (the light within the rear food alcove) to be illuminated when the subject is required to respond there.
- The trial then proceeds to an **INITIAL PAUSE** before the target stimulus is presented. You specify the length of the initial pause, in milliseconds (ms). You can specify one initial pause (e.g. 1000 ms) or a set of possible initial pauses (e.g. 500, 1000, 1500, and 2000 ms). By default the program picks an initial pause duration *at random* from this list. You can also choose **pseudorandom** selection, specifying a "DWOR multiplier" (see drawing without replacement). To specify values for the initial pause, click "Set possible values for initial pause":



Every time you click "Enter another value", your previous value is added to the list (shown near the top) and you can enter another. Click "I've finished" when you've finished entering the possible values you want. **Note also** that you can enter values twice, influencing their likelihood of selection. For example, if you enter "1000, 2000, 2000", then you will get initial pauses of 1000 or 2000 ms, but you will be twice as likely to get a 2000 ms initial pause than a 1000 ms pause.

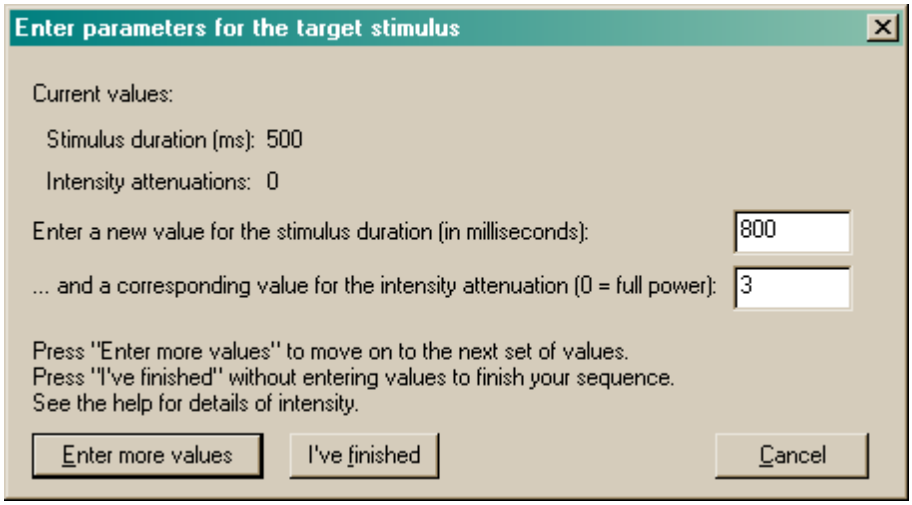
Example. *Suppose you want four equiprobable initial pause values (1500, 2000, 2500, and 3000 ms). Suppose you want 200 trials, and you want each initial pause value to be used exactly 50 times, but as randomly intermixed among those four trials as possible. Enter these four values into your list. Tick the "pseudorandom" option. Then enter "50" into the multiplier box.*

Example. *Suppose you want four equiprobable initial pause values (1500, 2000, 2500, and 3000 ms). Suppose you want 200 trials, and you want each initial pause value to be used exactly 50 times, with every initial pause value used once every four trials. Enter these four values into your list. Tick the "pseudorandom" option. Then enter "1" into the multiplier box.*

Example. *Suppose you want four equiprobable initial pause values (1500, 2000, 2500, and*

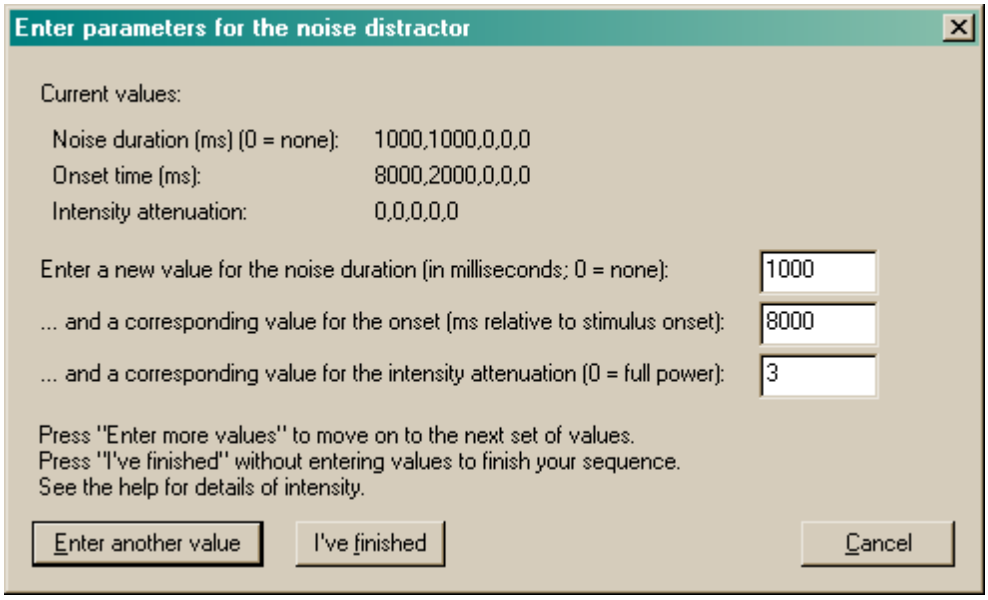
3000 ms). Suppose you want 200 trials, and you want each initial pause value to be used exactly 50 times, with every initial pause value used five times every twenty trials. Enter these four values into your list. Tick the "pseudorandom" option. Then enter "5" into the multiplier box.
Example. *Suppose you want two initial pause values (1500 and 2000 ms). Suppose you want your initial pause values to be chosen so that the probability of 1500 ms is 1/3, and the probability of 2000 ms is 2/3, but you would like a truly random selection otherwise. Enter "1500, 2000, 2000" into your list of values. Do not tick the "pseudorandom" option.*

- After the initial pause, the **STIMULUS** is presented. You can specify the stimulus parameters in exactly the same way as the initial pause, except that the numbers come in pairs: you specify a **duration** and an **intensity attenuation** where 0 represents full power. Your equipment may or may not support values other than 0 (full power) for the intensity attenuation (see dimming systems). You must specify parameters for at least one stimulus. A dialogue, similar to that for specifying the initial pause, is shown when you click "Set stimulus parameters":



  - On a given trial, the stimulus parameters are picked at random from the set of (paired) values. Note therefore that you can enter several copies of the same parameters: for example, if you wanted all your stimuli to be at full power, but you wanted on average 75% of them to last 500 ms and 25% to last 1000 ms, then you could enter the following (duration, intensity) pairs: (500, 0); (500, 0); (500, 0); (1000, 0).
  - There is also an option to draw the stimulus parameters **pseudorandomly** from the set you specify, and to specify a "DWOR multiplier" for this process (see drawing without replacement).

- You can specify the **LIMITED HOLD** period. This is the time the program will wait after the stimulus *begins* before it abandons the trial, scoring it as an omission, if the subject has not responded.
- You can also specify an optional **NOISE DISTRACTOR**. You do not have to specify any at all. If you would like to use them, you specify the noise duration (in ms), noise onset time, and intensity attenuation (just as for the stimulus). You can use a duration of 0 ms to mean "no noise"; this allows you to mix noise and non-noise trials. Noise onset times are relative to the start of the stimulus, and may be negative (if you want the noise distractor to occur before the stimulus) or positive (if you want the noise to come on after the stimulus begins) or zero (for simultaneous noise/stimulus onset). You specify the parameters for each noise option in the dialogue you'll see when you click "Set noise parameters":



- On a given trial, the noise parameters are picked at random from the set of (triplet) values. Note therefore that you can enter several copies of the same parameters: for example, if you wanted all your noises to be at full power, but you wanted half the trials to be without noise, and of the noise trials you wanted on average 50% of the noises to last 600 ms and 50% to last 8000 ms, and of the long stimuli you wanted 2/3 to have an onset time of -100 ms (100 ms before the stimulus begins) and 1/3 to be at +100 ms, and similarly for the short stimuli, then you could enter the following (duration, onset, intensity) triplets: (0, 0, 0); (0, 0, 0); (0, 0, 0); (0, 0, 0); (0, 0, 0); (0, 0, 0); (600, -100, 0); (600, -100, 0); (600, 100, 0); (800, -100, 0); (800, -100, 0); (800, 100, 0).
- There is also an option to draw the noise parameters **pseudorandomly** from the set you specify, and to specify a "DWOR multiplier" for this process (see drawing without replacement).

- Finally, task failure in a variety of ways (see Task design and Trial details) can lead to punishments in the form of **TIMEOUTS**. Specify the timeout duration.
- Front panel responses during timeouts are normally considered premature (if the timeout was induced by a premature response before a stimulus was presented) or perseverative (if the timeout was induced by incorrect responding after a stimulus was presented). The options "**Front panel responding in a pre-stimulus timeout is scored as premature**" and "**Front panel responding in a post-stimulus timeout is scored as perseverative**", both on by default, allow you to vary this behaviour. See the Trial details.
- Optionally (see Trial details), all responses at the front holes during a timeout can be punished, by prolonging (restarting) the timeout. Tick "**Front panel responding during timeout prolongs (restarts) the timeout**" to enable this feature. For details of how these responses are scored, see the Trial details.
- Similarly, if you wish, you can tick "**Front panel responding while waiting to start trial is punished**". For details, see Task design and Trial details. These responses are classified as premature; see Task design and Trial details.
- There is an option to "**Punish perseverative nosepokes following a correct response**", similarly. See Trial details for full details.

The last set of options concern miscellaneous aspects of the task.

- **Reward parameters.** Choose the number of pellets to be used as a reward, and the timing used for your particular brand of pellet dispenser (e.g. if you would like 2 pellets per reward, delivered 500 ms apart, and your pellet dispenser likes a 45-ms electrical pulse to dispense a single pellet, then specify 2, 45, and 500 in the boxes here).
- **Debouncing.** Electromechanical devices often "bounce": if you press a lever, a short burst of

electrical pulses are sent, as physical vibration in the lever turns the device on and off very rapidly. You normally want to ignore this. Typically, you might set a "debounce time" of 10 ms; this would mean that any responses that are repeated within 10 ms of a previous response on the same device (e.g. lever, panel detector) are attributed to electromechanical bounce and ignored. Mammals generally cannot make a voluntary action twice within 10 ms! If you don't debounce inputs, strange things can happen; for example, we saw a problem as a result of a mechanical food alcove switch bouncing; this registered erroneously as perseverative rear panel pushes when it was really a single trial-starting response.

- **Ignore all line OFF events.** If ticked, the program ignores all "off" events (e.g. the subject *ceasing* to respond at the rear panel). Actually, in the current version of the task, these events are ignored anyway, but the task asks the Whisker server about them, so this option just disables some warning messages in systems that can't detect lines going off as well as on! In future versions of the task, the "off" events may be recorded if this option is not ticked.
- **Intensity attenuation (dimming) system.** Specify the type of dimming system (q.v.) that your equipment supports. If you select the "new" dimming system, you can also specify the time that the DIM lines should be left alone after a device is switched on (see dimming systems). This parameter constrains your ability to specify stimuli and noises that overlap too closely if they are of different intensities.

1.9 Randomness, pseudorandomness, drawing without replacement

Suppose I wish to pick a series of numbers from 1 to 6.

I could pick them **at random**. I could do this by rolling a true die. Every time I rolled the die, I would obtain a number from 1-6 with equal probability. The probability of obtaining a "1" would be 1/6 (approximately 0.17); the probability of obtainin a "2" would be 1/6, and so on.

If I rolled the die 100 times, I would expect to get roughly 17 ones, roughly 17 twos, roughly 17 threes, roughly 17 fours, roughly 17 fives, and roughly 17 sixes. It is *possible* that I would get 100 sixes—but very unlikely (the probability of this is 1.53×10^{-78}, or one in six hundred thousand quintillion quintillion quintillion quintillion). But it is certain that I would not get exactly the same number of ones, twos, threes, fours, fives, and sixes (because you can't have six equal whole numbers adding up to 100). If I rolled the die 60 times, you'd expect about 10 of each number—but it's also pretty unlikely that I'd get *exactly* 10 of each number.

If I rolled the die like this, there is absolutely no way to predict the next number that will come up on each roll.

So much for randomness.

If I want a series of 60 numbers and I want to ensure that I end up with equal numbers of ones, twos, threes, fours, fives, and sixes, I could simply take them in a **predictable order**: 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6. This gives me the distribution I want (ten of each number), but the sequence is anything but random, and is highly predictable. If you know the last number that came up, you have an extremely good idea of what the next number would be.

So much for total predictability.

To obtain a reasonable combination of unpredictability and obtaining an overall equal distribution of numbers, we could use a **pseudorandom** technique called **drawing without replacement**, or **draw-without-replacement**, sometimes shortened to **draw-w/o-replacement** or just **DWOR**. Imagine we would like a sequence of 60 numbers, each in the range 1-6, more or less randomly, but ensuring that we get 10 of each number. We could write the number "1" on ten pieces of paper, the number "2" on ten more pieces of paper, and so on. We could then put all 60 pieces of paper in a

hat, shuffle them, and draw them out in sequence, *without replacing them*. We'd then be guaranteed ten of each type, but the local order would be fairly random. It would not be totally random—if we've drawn out 10 ones, 10 twos, 9 threes, 10 fours, 10 fives, and 10 sixes, then we can be absolutely certain that the last number out of the hat will be a 3. However, we'd have to remember the numbers that had come out so far.

So it's impossible to have a completely random order and to guarantee a certain distribution—but drawing without replacement is a good way to approximate randomness while guaranteeing a certain distribution.

There are several ways to draw without replacement. I've just described a situation in which 10 copies of each number (1-6) are put into the hat, shuffled, and drawn individually for 60 trials. It's possible (but again extremely unlikely!) that the first ten trials would all be ones, the next ten all twos, and so on. If we wanted to guarantee that *in every six consecutive trials* we will see each possible digit (1-6) once, we should do something different. We should write the number "1" on one piece of paper, the number "2" on a second piece of paper, and so on, up to six pieces of paper. We should then put the six into the hat, shuffle them, and draw them out (without replacement) for the first six trials. We should then put the six pieces of paper back in the hat, shuffle, and repeat the process for the next six trials, and so on. This process gives *less randomness* (if you know just the first five trials in a set of six, then in principle, you can have perfect knowledge of the sixth number to be drawn) but *better control over the local distribution* of numbers.

We've just seen two examples in which a **list** of six numbers (1, 2, 3, 4, 5, 6) are put into a hat and drawn for 60 trials. In the first, we put ten copies of each item in the list into the hat (giving 60 pieces of paper in total) and drew them. I call this a **draw-without-replacement (DWOR) multiplier** of 10. In the second, we put one copy of each item in the list into the hat, drew them until we'd run out of numbers (after six trials), then put them all back into the hat. I call this a DWOR multiplier of 1.

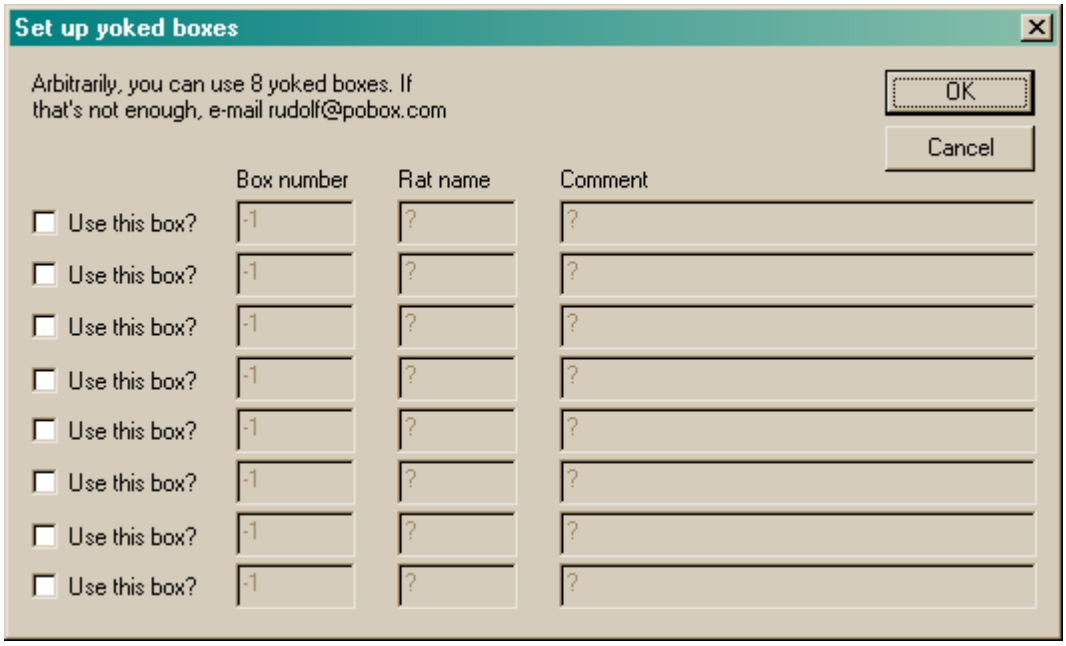
We've just seen the concept of a *list* of possibilities, which is multiplied by a *DWOR multiplier* to give a set of options that are then *drawn at random without replacement* from a hat; when the hat is empty, we restart the process.

The bigger the DWOR multiplier, the closer the DWOR technique comes to total randomness (if the DWOR multiplier were infinitely large, they are exactly the same process). The smaller the DWOR multiplier, the closer the technique is to total predictability.

This program offers the DWOR technique as a way of selecting possible values for various parameters (such as stimulus durations).

1.10 Yoking

The Yoking dialogue box looks like this:



In a **yoked** experiment, yoked (slave) animals receive stimuli and rewards not according to their behaviour, but according to the behaviour of another (master) rat.

When you run a single copy of the FiveChoice task, **one** rat can act as a master rat. If you are not interested in yoking, then every rat is its own master rat and there are no yoked (slave) animals.

Here, you may set up up to 8 yoked animals that will experience exactly the same stimuli as the master rat (the one you configure in the normal way).

Data recording for yoked (slave) subjects

First refer to the trial format for master rats and ensure you understand it. Slave recording is detailed below. Note that **nothing a yoked subject does has any behavioural consequences.** Red text indicates responses not possible for master rats. Note that some odd sequences are possible: for example, it's even possible for the yoked rat to collect reward before making a correct/incorrect response.

State name	Description	State of the box	Consequence of nosepoking at the front panel (one of the five holes)	Consequence of nosepoking at the rear (food) panel	Consequence of time passing
NOTSTARTED	Not started yet. Awaiting experimenter to start task.	Darkness.	-	-	-
PRESTIM_PLEASEPUSH	**Task begins here.** (For the first trial, a free pellet is given and the trial number is initially set to 0.) Rat must respond	Houselight on. Traylight on (if used).	Score **premature response.** No other consequence.	Recorded as attempt to start new trial (and many such responses may potentially be	-

	to rear panel to initiate the trial.			recorded).	
POSTSTIM_PLEASEPUSH	Post-stimulus punishment timeout over. Rat must respond to rear panel to initiate new trial.	Houselight on. Traylight on (if used).	Score **perseverative response.** No other consequence.	Recorded as attempt to start new trial (and many such responses may potentially be recorded).	-
INITIAL_PAUSE	**Trials begin here. Trial number incremented.** Variable time before stimulus is presented (sometimes called the "ITI").	Houselight on.	Score **premature response.** No other consequence.	Score **perseverative panel-push.** No other consequence.	-
STIM_ON	Stimulus presented briefly. Opportunity to respond.	Houselight on. One target hole lit.	If this is the **first** front panel response since stimulus presentation, score as **correct** or **incorrect. Second and subsequent** responses are scored as **perseverative.**	Score **perseverative panel-push.** No other consequence.	If the yoked rat has not responded to the front panel within the limited hold period, or before the next trial starts, score an **omission.**
STIM_OFF	Stimulus has gone off, but opportunity to respond remains.	Houselight on.	If this is the **first** front panel response since stimulus presentation, score as **correct** or **incorrect. Second and subsequent** responses are scored as **perseverative.**	Score **perseverative panel-push.** No other consequence.	If the yoked rat has not responded to the front panel within the limited hold period, or before the next trial starts, score an **omission.**
AWAITING_COLLECT	Successful response; subject has been rewarded. Awaiting food collection (which will also initiate the next trial).	Houselight on. Traylight on (if used).	If this is the **first** front panel response since stimulus presentation, score as **correct** or **incorrect. Second and subsequent** responses are scored as **perseverative.**	**First** such response: score as reward collection for latency purposes. **Second and subsequent** responses are recorded individually.	If the yoked rat has not responded to the front panel within the limited hold period, or before the next trial starts, score an **omission.**
PRESTIM_TIMEOUT	Rat is being punished for a premature nosepoke to the front panel.	Darkness.	Score **premature response.** No other consequence.	Recorded. No other consequence.	-

POSTSTIM_TIMEOUT	Rat is being punished for an incorrect response, or a perseverative response.	Darkness.	If this is the **first** front panel response since stimulus presentation, score as **correct** or **incorrect. Second and subsequent** responses are scored as **perseverative.**	Recorded. No other consequence.	If the yoked rat has not responded to the front panel within the limited hold period, or before the next trial starts, score an **omission.**
FINISHED	Rat finished the task	Darkness.	-	-	-
ABORTED	User aborted the task	Darkness.	-	-	-

1.11 Results

The program always stores results in two places. One is a human-readable text file. The other is a database. (You choose the name of this file in the main parameters dialogue box, and you can choose the database here as well.)

1.11.1 Text-based results file

A sample results file is shown below. The configuration information is shown first; the results follow. (There aren't very many results, because I got bored creating the file.) The results section is shown in bold, with **trial summary** information followed by **individual response** information.

I encourage you to think of this file as a backup. The database contains all this information and can be used to retrieve both simple and highly detailed information about a subject's performance.

```
FIVE-CHOICE SERIAL REACTION TIME TASK -- SUMMARY FILE
_____________________________________________________

FiveChoice v3.2 - release build compiled on Apr  5 2006 at 10:53:58

IDENTIFICATION

Rat:                    subject2
Session:                56
Date/time code:         05-Apr-2006 (10:54)
Comment:                (add your comment here)

Box:                    0
Client computer name:   EGRET
Server computer name:   loopback
Summary file name:      D:\Whisker\CODE\clients\rnc - cambridge\FiveChoice\junk-
configs\subject2-05Apr2006-1054-FiveChoice-summary.txt
Default ODBC database: FiveChoice_prototype
Preferred box number:   0

CONFIGURATION
```

```
Target number of trials                 : 10
Max trials of all types (0=no limit)    : 20
Session time limit (min)                : 20
Use traylight?                          : Y
Initial pause values (ms)               : 500,1000,1500,2000
Pseudorandom initial pause selection?   : N
... draw-w/o-replacement multiplier     : 50
Stimulus duration values (ms)           : 500
Pseudorandom stimulus duration selection?: N
... draw-w/o-replacement multiplier     : 50
Limited hold duration (ms)              : 5000
Noise duration (ms, 0=none)             : 0
Noise onset (ms after stimulus)         :
Minimum timeout duration (ms)           : 5000
Number of pellets                       : 1
Pellet pulsing time (ms)                : 40
Interpellet gap (ms)                    : 150
Front panel prolongs timeouts           : Y
Front punished while waiting to start   : N
Punish 'perserverative' after 'correct'? : Y
Input debounce time (ms)                : 10
Forced-choice task?                     : N
... If so, forced hole is               : 4
Pseudorandom target location selection?  : N
... draw-w/o-replacement multiplier     : 1
Ignore all line OFF events              : Y

SUMMARY DATA

Started at:        05-Apr-2006 (10:54)
Finished at:       05-Apr-2006 (10:55)

TOTALS (master box)

Number of trials                : 7
Number of correct responses     : 3
Number of incorrect responses   : 2
Number of omissions             : 0
Number of 'valid' trials        : 5
Number of pellets earned (total) : 4

YOKING CONFIGURATION

Rat,Box,Yoked,MasterRat,MasterBox,Comment
subject2,0,N,subject2,0,(add your comment here)

TRIAL AND RESPONSE DATA

Rat,Box,Trial,SessionBeganAt_ms,TrialBeganAt_ms,PleasePushBeganAt_ms,
InitialPauseBeganAt_ms,StimulusOnAt_ms,RewardedAt_ms,TimeoutBeganAt_ms,
PrematureNosepokes,PerseverativePanelPushes,PerseverativeNosepokesSameHole,
PerseverativeNosepokesOtherHoles,InitialPauseDuration_ms,NoiseOnsetTimeRelStim_ms,
OfferedHole,ChosenHole,ResponseLatency_ms,ExperiencedTimeout_ms,
CollectionLatency_ms,Correct,Incorrect,Omission,IntendedStimulusDuration_ms,
StimulusOffAt_ms
subject2,0,1,3328985331,3328995007,3329003623,3328995007,3328996508,4294967295,332
8998619,0,0,0,0,0,1500,-1,1,2,2096,5000,4294967295,0,1,0,500,3328998610
subject2,0,2,3328985331,3329004623,4294967295,3329004623,3329006123,3329008064,429
4967295,0,0,0,0,0,1500,-1,4,4,1930,4294967295,2414,1,0,0,500,3329008059
subject2,0,3,3328985331,3329010487,4294967295,3329010487,3329011487,3329012109,429
4967295,0,0,0,0,0,1000,-1,0,0,611,4294967295,3569,1,0,0,500,3329012105
subject2,0,4,3328985331,3329015686,3329023177,3329015686,3329016697,4294967295,332
```

```
9018173,0,0,0,0,0,1000,-1,4,1,1462,5001,4294967295,0,1,0,500,3329018165
subject2,0,5,3328985331,3329024134,3329033307,3329024134,4294967295,4294967295,332
9024568,7,0,0,0,0,1500,-1,-1,-
1,4294967295,8736,4294967295,0,0,0,4294967295,4294967295
subject2,0,6,3328985331,3329033953,4294967295,3329033953,3329034953,3329037979,429
4967295,0,0,0,0,0,1000,-1,3,3,3016,4294967295,2151,1,0,0,500,3329037975
subject2,0,7,3328985331,3329040139,4294967295,3329040139,3329040639,4294967295,429
4967295,0,0,0,0,0,500,-1,2,-
1,4294967295,4294967295,4294967295,0,0,0,500,3329041144

Rat,Box,Trial,ResponseNum,Location,Phase,TimeAbsolute_ms,TimeInSession_ms,
TimeInTrial_ms,InterestingTime_ms,ResponseType
subject2,0,1,0,2,6,3328998604,13273,3597,2096,4
subject2,0,1,1,99,3,3329004614,19283,9607,5995,5
subject2,0,2,2,4,6,3329008053,22722,3430,1930,3
subject2,0,2,3,99,7,3329010478,25147,5855,2414,7
subject2,0,3,4,0,6,3329012098,26767,1611,611,3
subject2,0,3,5,99,7,3329015678,30347,5191,3569,7
subject2,0,4,6,1,6,3329018159,32828,2473,1462,4
subject2,0,4,7,99,3,3329024127,38796,8441,5954,5
subject2,0,5,8,0,4,3329024559,39228,425,425,1
subject2,0,5,9,0,8,3329024825,39494,691,257,1
subject2,0,5,10,0,8,3329025262,39931,1128,694,1
subject2,0,5,11,0,8,3329025715,40384,1581,1147,1
subject2,0,5,12,0,8,3329026064,40733,1930,1496,1
subject2,0,5,13,0,8,3329026634,41303,2500,2066,1
subject2,0,5,14,99,8,3329027392,42061,3258,2824,8
subject2,0,5,15,2,8,3329028303,42972,4169,3735,1
subject2,0,5,16,99,8,3329029323,43992,5189,4755,8
subject2,0,5,17,99,2,3329033945,48614,9811,9377,5
subject2,0,6,18,3,6,3329037969,52638,4016,3016,3
subject2,0,6,19,99,7,3329040130,54799,6177,2151,7


Successfully wrote to database: ODBC;DSN=FiveChoice_prototype;DBQ=D:
\Whisker\CODE\clients\rnc - cambridge\FiveChoice\FiveChoice database (prototype).
mdb;DriverId=281;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;
```

1.11.2 Creating a new ODBC source

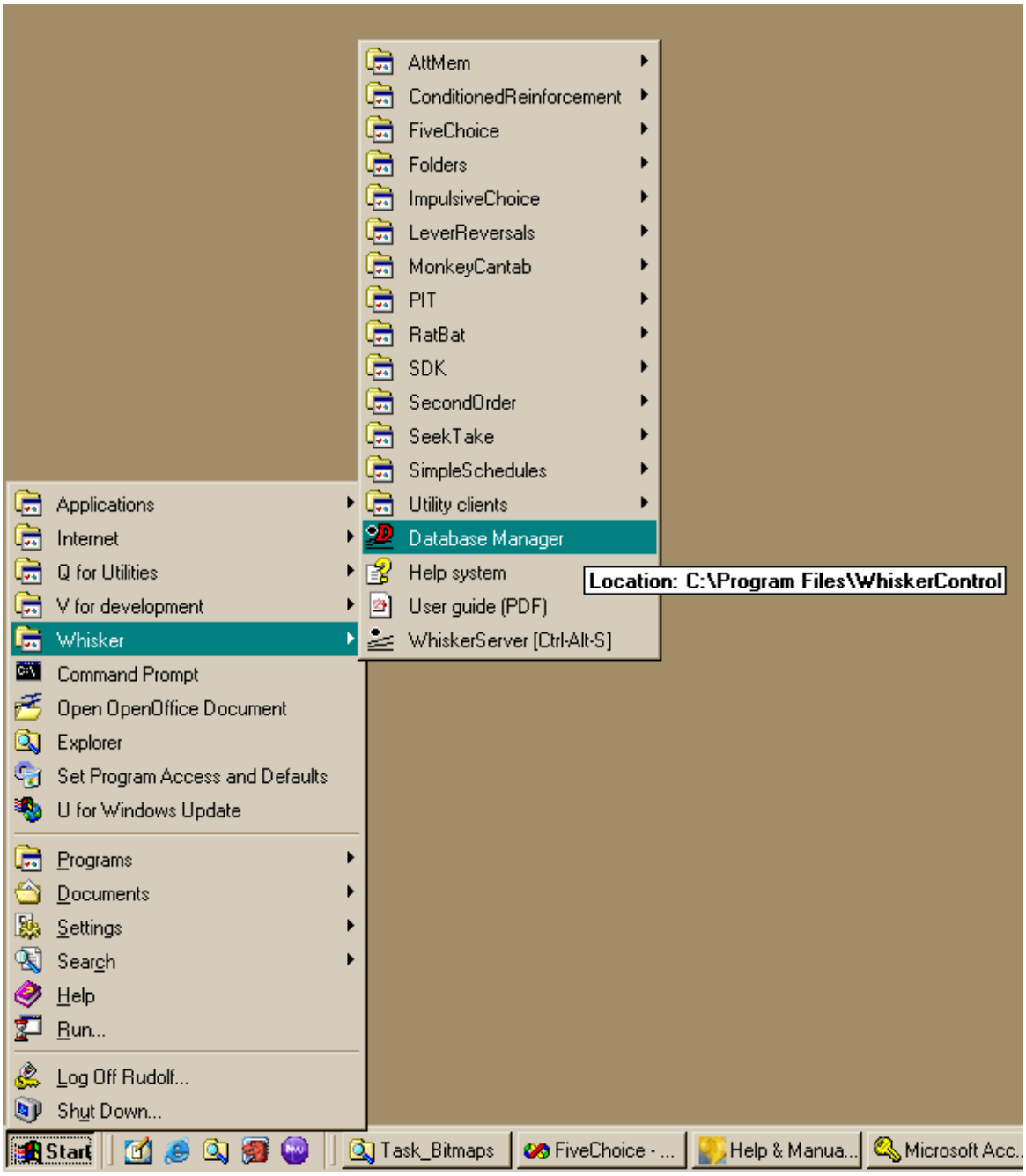
What happens if you're using FiveChoice for the first time, or you're starting a new experiment, and you need to set up a new ODBC (Open Database Connectivity) source for FiveChoice? You should configure it via the **Whisker Database Manager** (Start → Whisker → Database Manager), or via **Control Panel → ODBC** [in Windows 2000, Start → Settings → Control Panel → Adminstrative Tools → Data Sources (ODBC)]. This section shows you various ways to achieve this.

Remember: you shouldn't use the supplied database without making a copy for yourself. (It will work, but if you ever uninstalled or reinstalled MonkeyCantab, this file might be replaced or lost. It is much safer to make your own copy and set up ODBC to use your copy.)

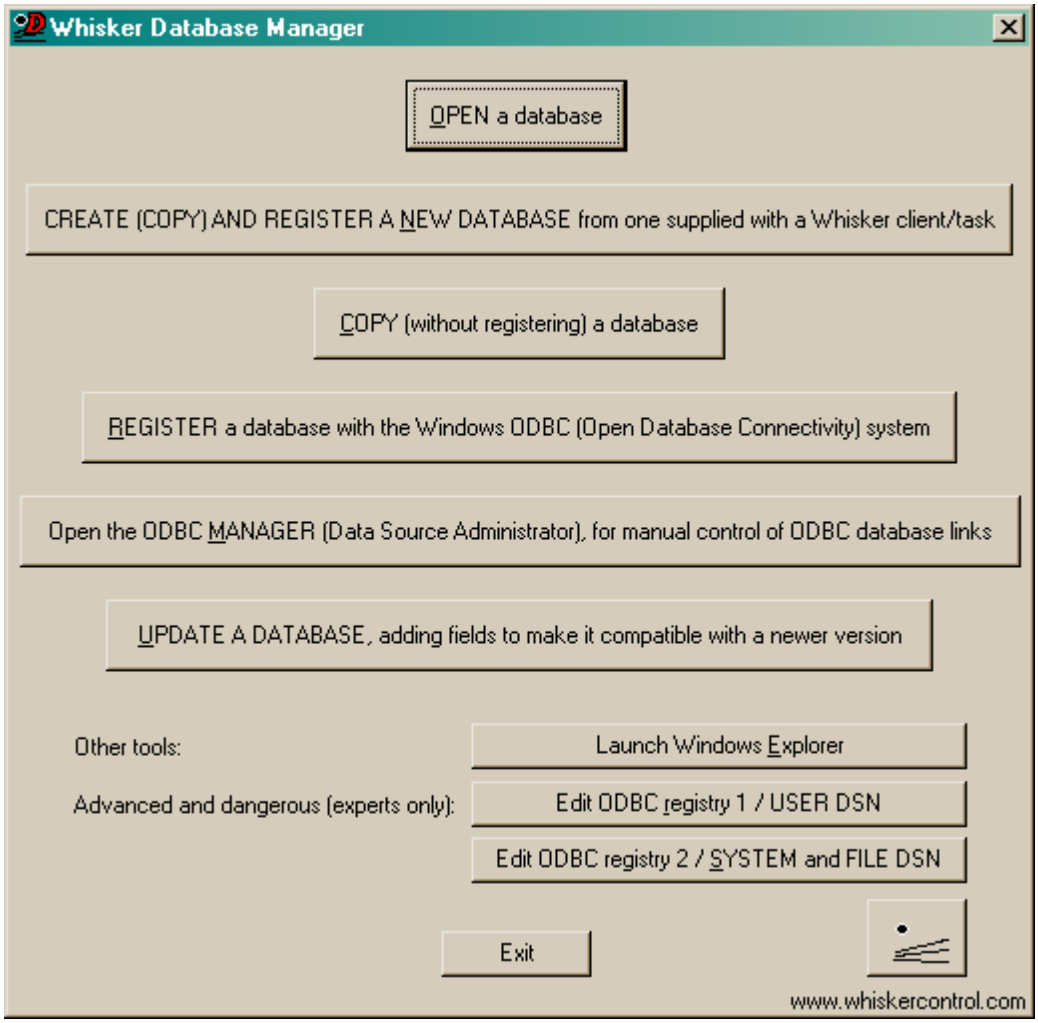
The procedure is:

1. **Make a copy of the supplied database to store *your* data in.**
2. **Register *your* copy with ODBC.**

The simplest way is to run the **Whisker Database Manager**:



You'll see the Database Manager:

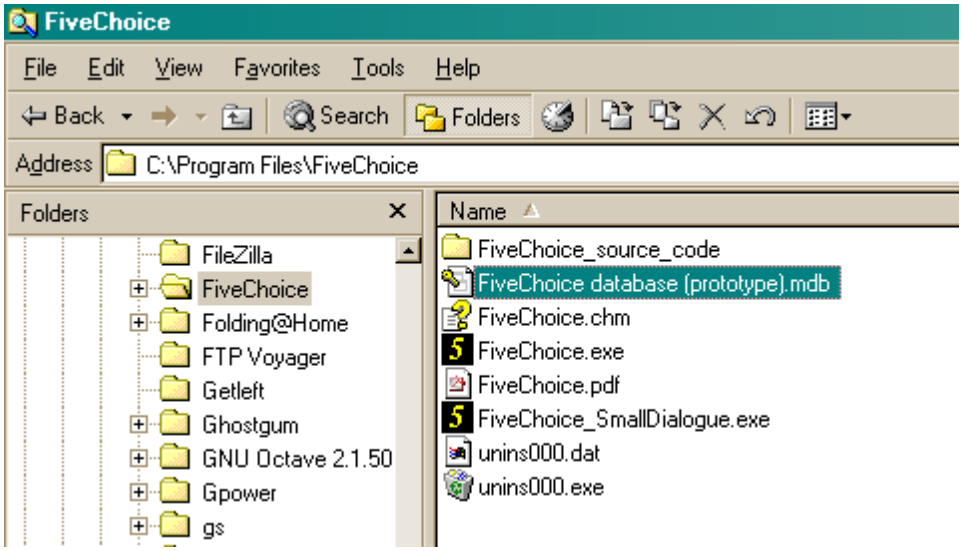


You can use this program to **CREATE (COPY) AND REGISTER** a database. Or you can **COPY** the supplied database and **REGISTER** your copy with ODBC separately. For full details of the Whisker Database Manager, see the **Whisker Help (Auxiliary Programs / Whisker Database Manager)**.

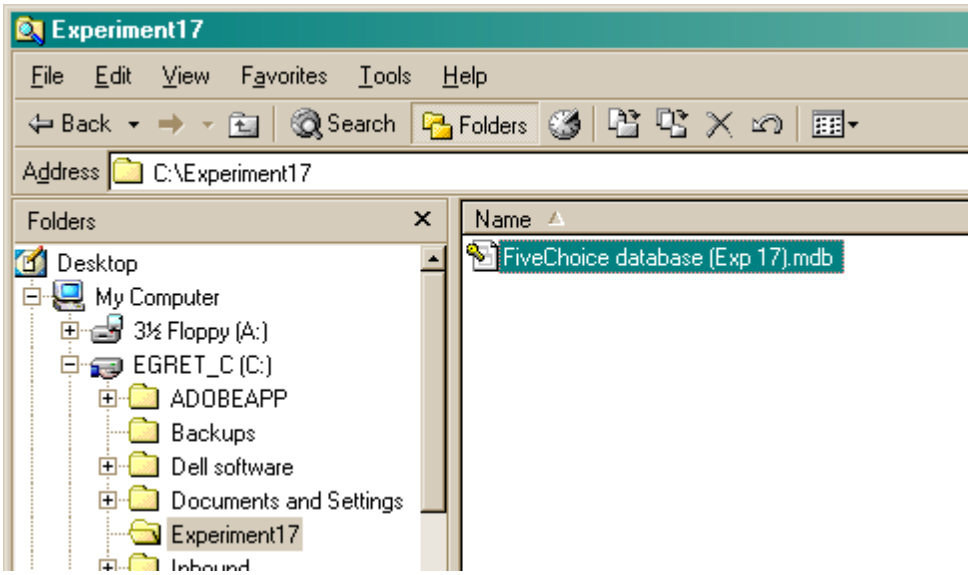
You can also do the whole thing by hand.

STEP 1.

First, make a copy of the supplied database to store *your* data in. Copy the supplied database:



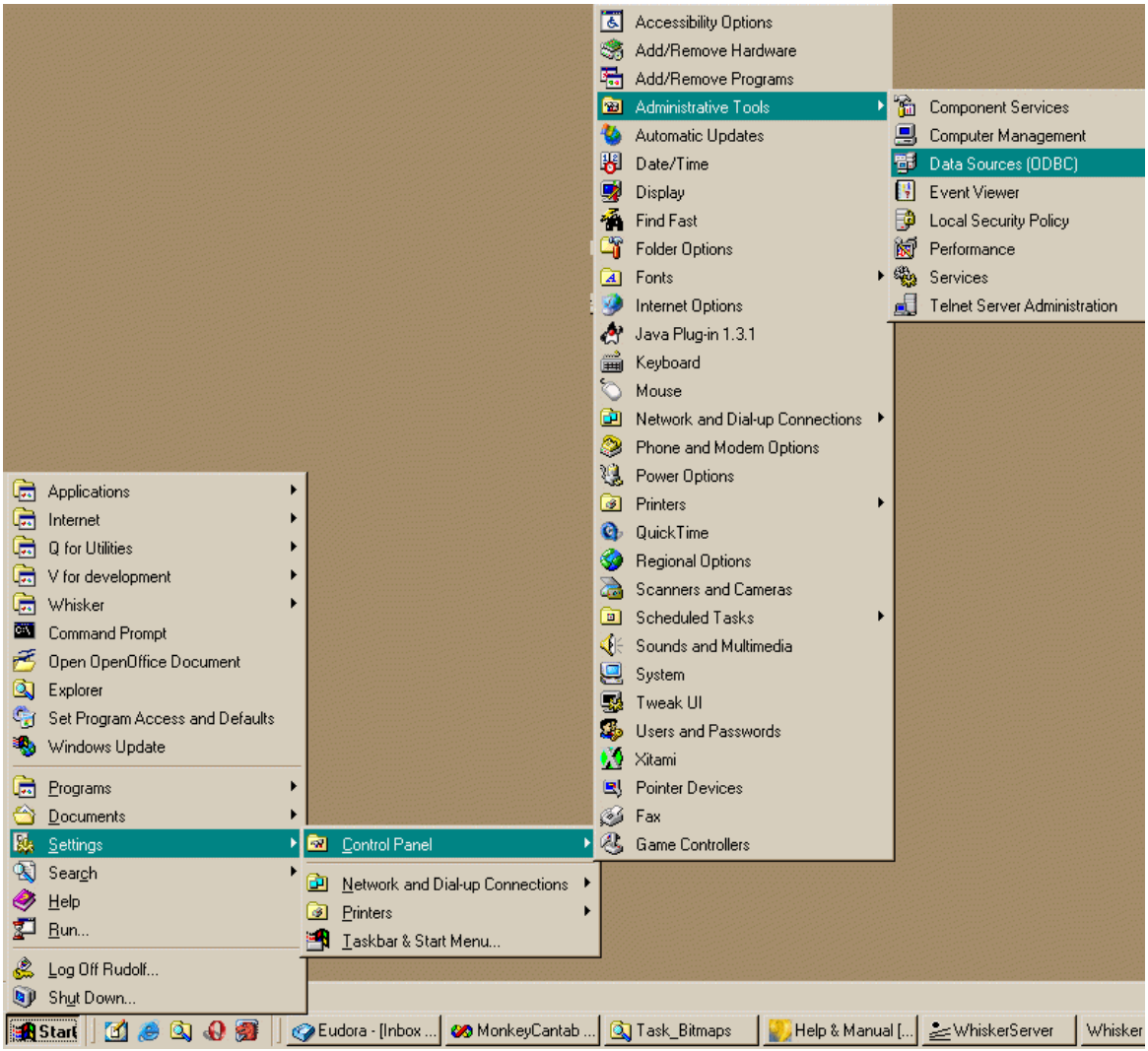
to your own:



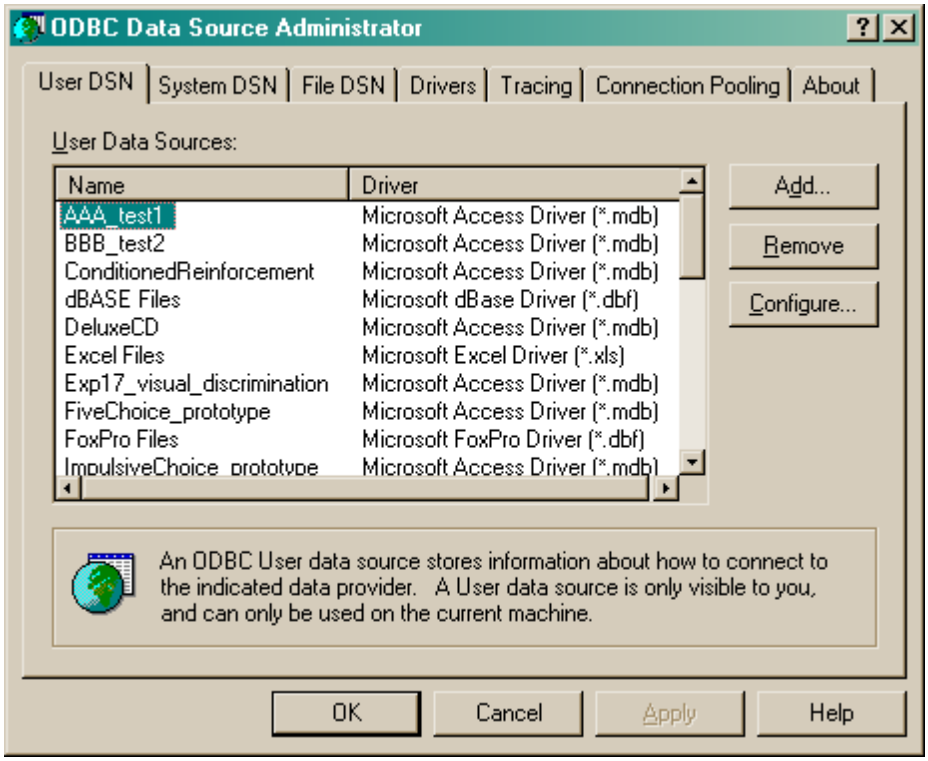
STEP 2.

Now, having already made a working copy of the prototype database supplied with the task, as described above, set *your* copy up as an **ODBC source** as follows.

Choose **Control Panel** → **ODBC** [in Windows 2000, **Control Panel** → **Adminstrative Tools** → **Data Sources (ODBC)**, or the equivalent for your version of Windows:

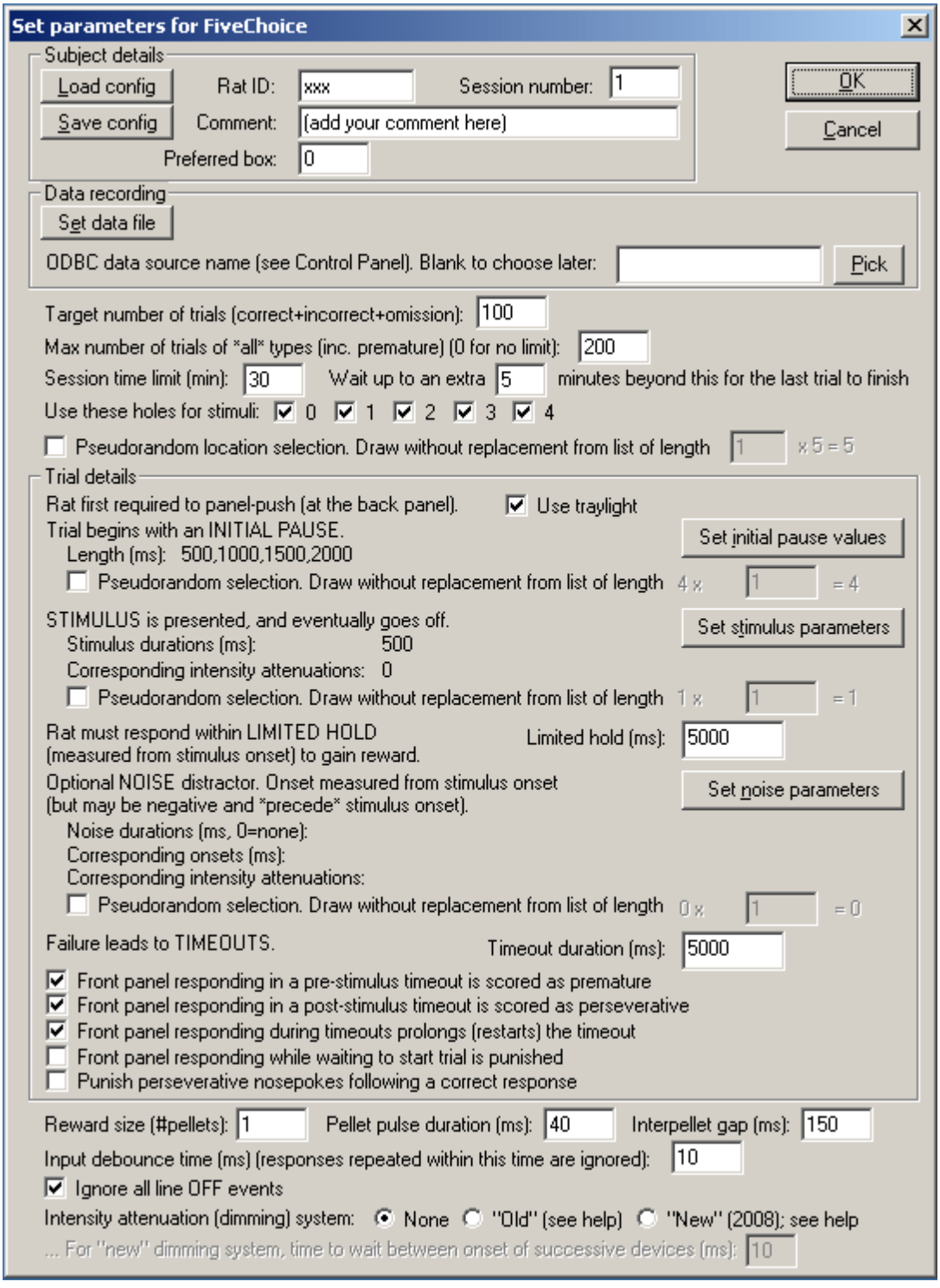


You'll see this:

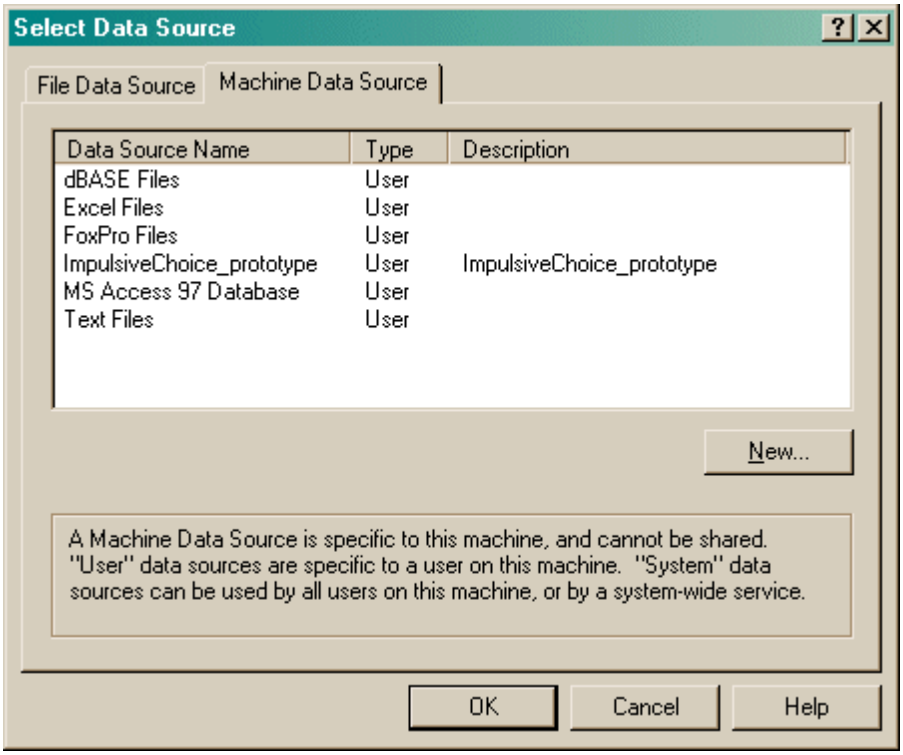


Click **Add**.

Alternatively, you can get to the same point from FiveChoice itself. Click **Pick** from the Parameters dialogue:

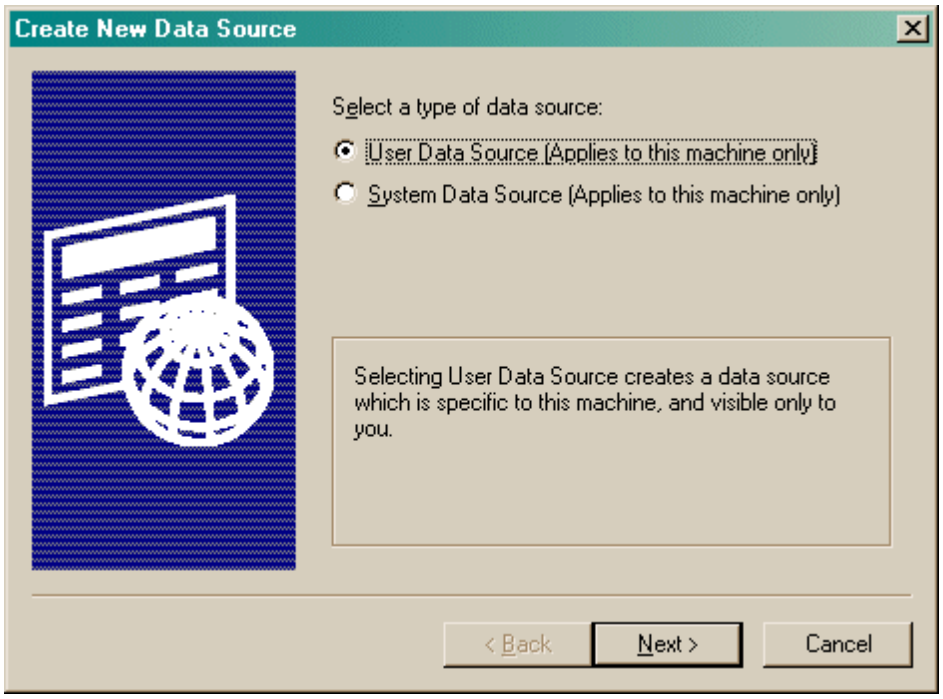


You'll see this:

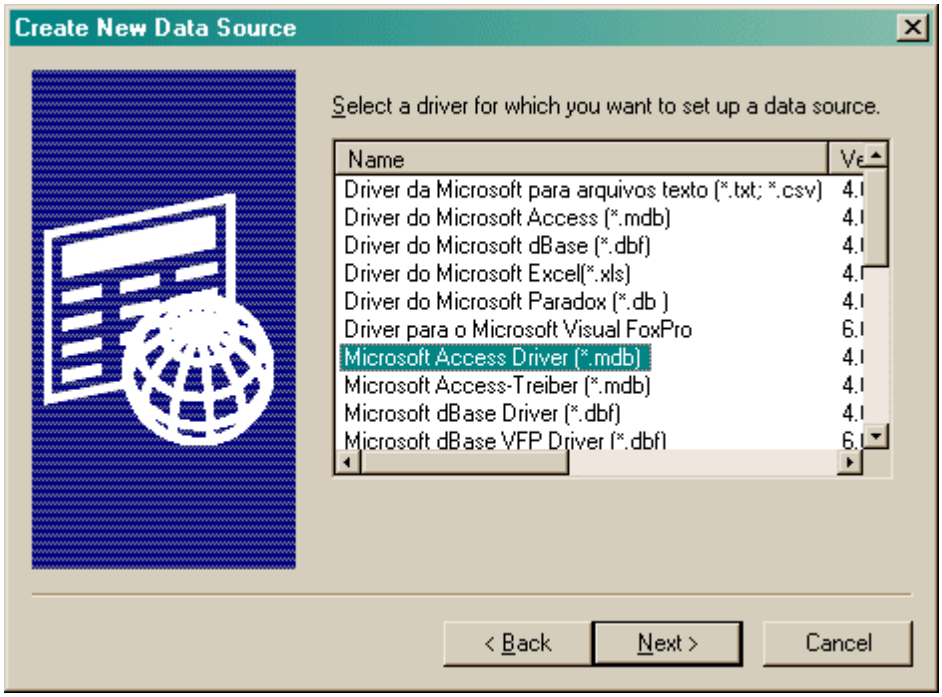


Click **New**.

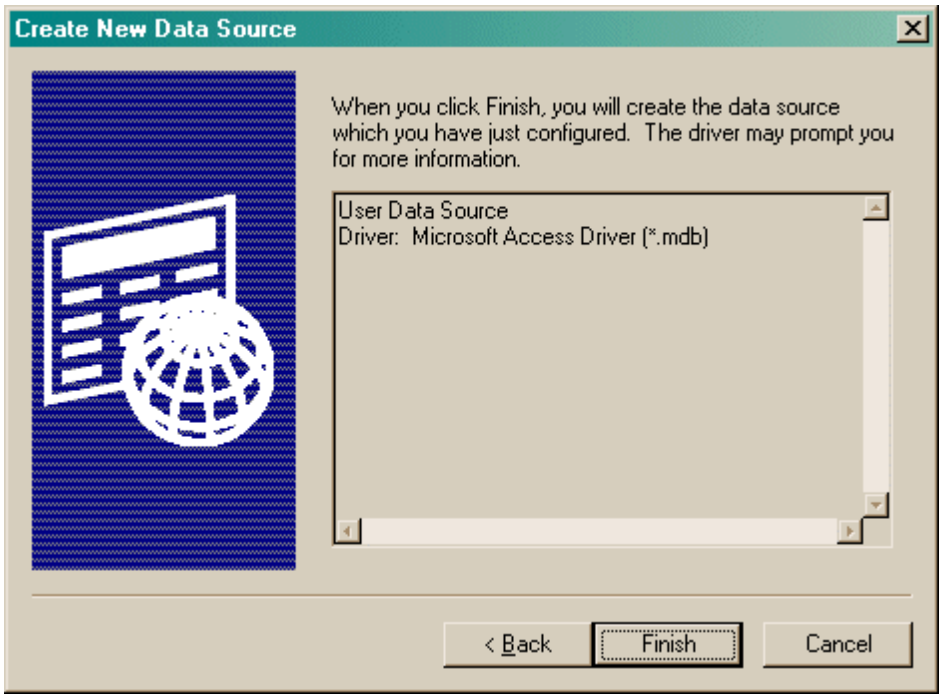
However you got here, you'll see something like this:



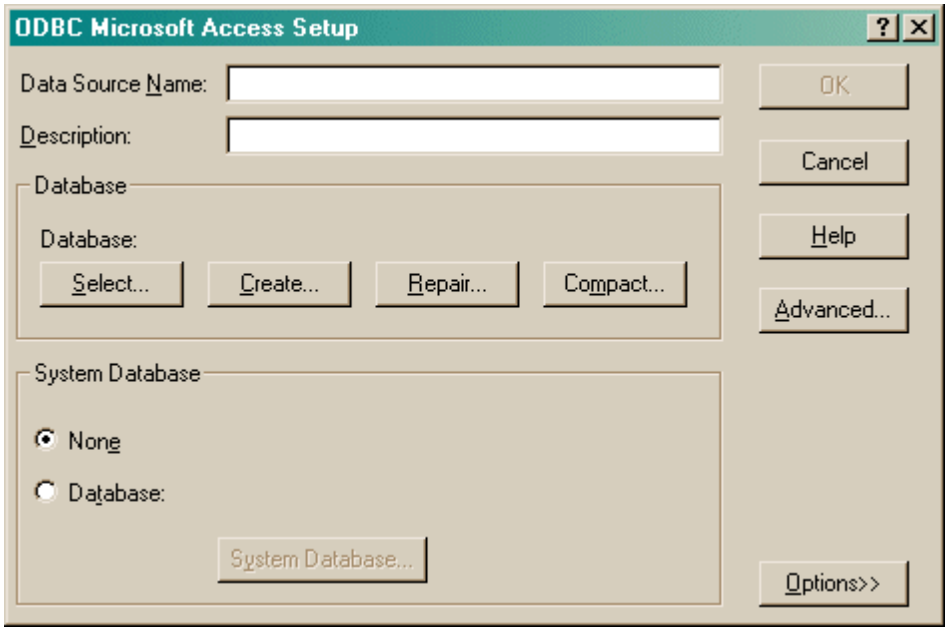
Choose a User or System data source. "User" databases are seen by people logged in as the current user. "System" databases are seen by anybody logged on to this computer. **User** is probably more sensible. Click Next.



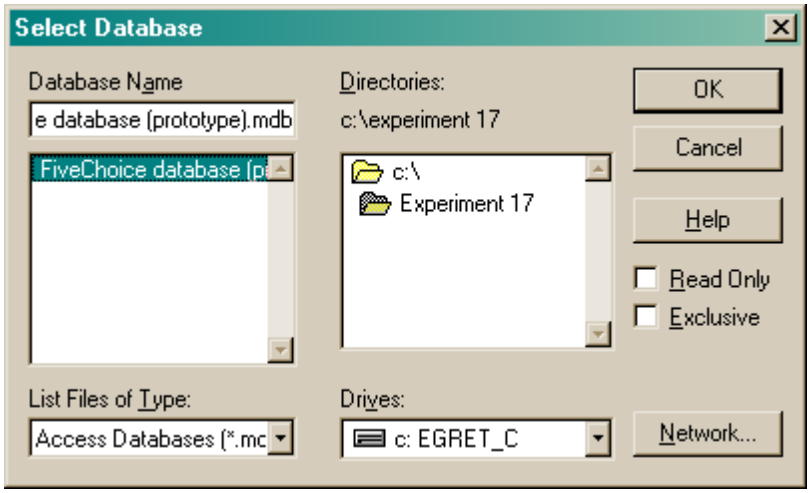
Choose your database driver. You probably want one that's in your language, and the supplied database is in Microsoft Access format (although MonkeyCantab itself will store data in any suitable ODBC-compatible database that has the right table and field names). Click Next.



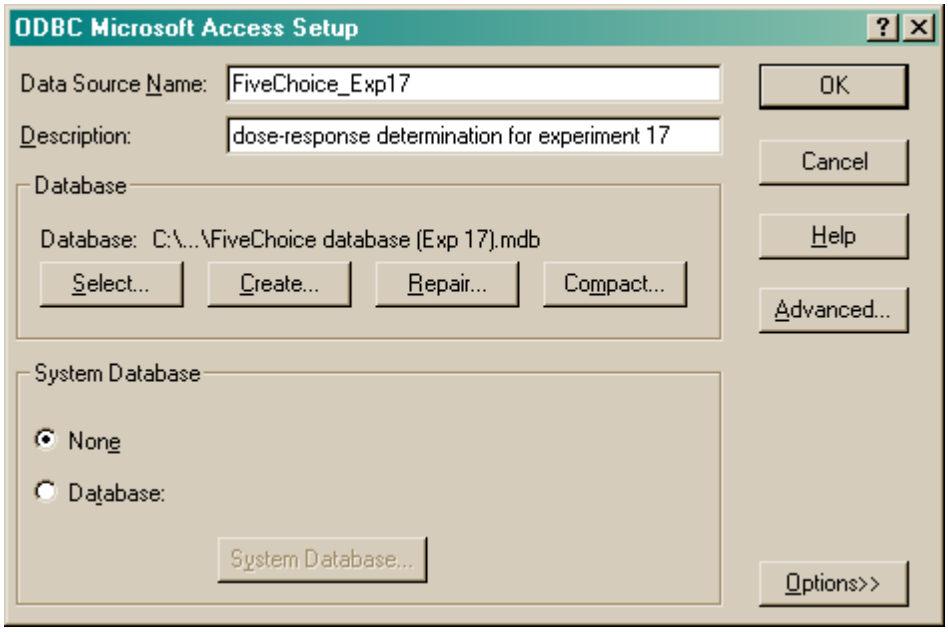
Click Finish. You'll see this:



You should fill in the **Data Source Name (no spaces)** and the **description,** and **Select** a database. When you click Select, this dialogue box appears:



Choose your database here and click OK. Your ODBC data source fields should now all be set up:



Click OK. You will be returned to the ODBC selection screen with your new data source now available.

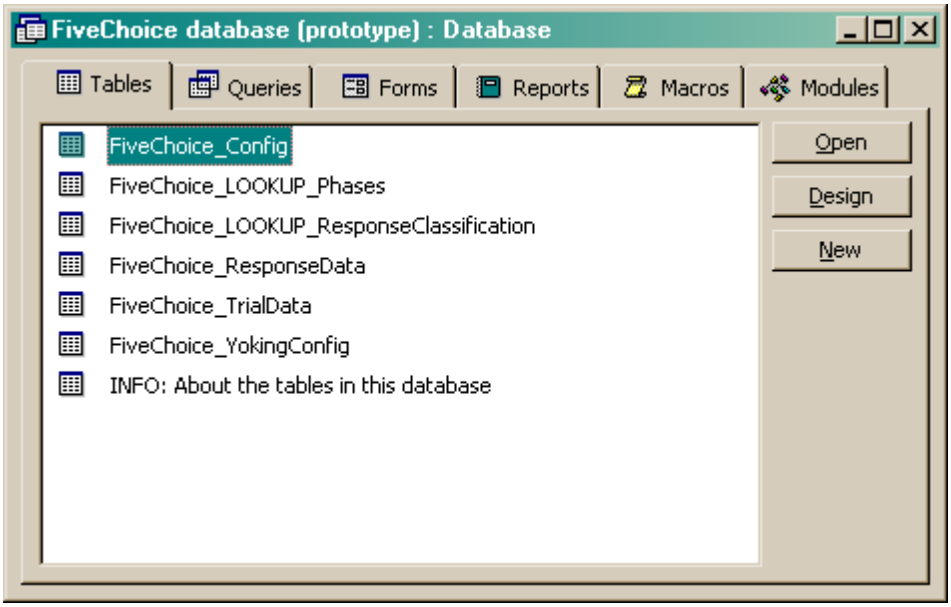
1.11.3 Using the Microsoft Access database for FiveChoice

Remember: you shouldn't use the supplied database without making a copy for yourself. (It will work, but if you ever uninstalled or reinstalled MonkeyCantab, this file might be replaced or lost. It is much safer to make your own copy and set up ODBC to use your copy. See Creating a new ODBC source.)

When supplied, the database is called **"FiveChoice database (sample).mdb"**. Make a copy before using it!

You need Microsoft Access (97 or higher) to use this database. Sorry about that.

When you open the database, it looks like this:



FiveChoice will store its results here. The table called "**INFO: About the tables...**" contains a description of each table (double-click it, or click it and click Open, to see the descriptions).. Click a table and click **Design** to view a list of all the fields. Here, for example, is the design view for the FiveChoice_TrialData table (which stores summary results for each trial):

FiveChoice_TrialData : Table

Field Name	Data Type	Description
RecordNum	AutoNumber	(Automatically generated.)
DateTimeCode	Date/Time	The date/time the task was started.
Rat	Text	Subject ID
Box	Number	Box number
Trial	Number	Trial number
SessionBeganAt_ms	Number	Time (by the system clock, in ms) the session started
TrialBeganAt_ms	Number	Time (by the system clock, in ms) the trial started
PleasePushBeganAt_ms	Number	Time (by the system clock, in ms) the subject was first able to initiate the trial by responding at the back panel
InitialPauseBeganAt_ms	Number	Time (by the system clock, in ms) the pause between initiation and the target stimulus started
StimulusOnAt_ms	Number	Time (by the system clock, in ms) the target stimulus was presented
RewardedAt_ms	Number	Time (by the system clock, in ms) the subject was rewarded (if it was rewarded!)
TimeoutBeganAt_ms	Number	Time (by the system clock, in ms) the timeout began, if one was given
PrematureNosepokes	Number	Number of premature nosepokes at the front holes
PerseverativePanelPushes	Number	Number of perseverative rear panel pushes
PerseverativeNosepokes	Number	Number of perseverative nosepokes at the front holes
PerseverativeNosepokesSameHole	Number	Number of perseverative responses to the same hole as the subject's first response
PerseverativeNosepokesOtherHoles	Number	Number of perseverative responses to holes other than the one the subject first responded at
InitialPauseDuration_ms	Number	Duration of the initial pause (ms)
NoiseOnsetTimeRelStim_ms	Number	Noise onset time, relative to stimulus onset time (ms)
OfferedHole	Number	Offered hole number (0-4)
ChosenHole	Number	Chosen hole number (0-4)
ResponseLatency_ms	Number	Response latency, in ms
ExperiencedTimeout_ms	Number	Duration of timeout experienced, in ms
CollectionLatency_ms	Number	Latency to collect reward, in ms
Correct	Yes/No	Was the response correct?
Incorrect	Yes/No	Was the response incorrect?
Omission	Yes/No	Was the trial scored as an omission?
IntendedStimulusDuration_ms	Number	How long was the intended stimulus duration? (The actual stimulus duration may be shorter than this, if the subje
StimulusOffAt_ms	Number	Time (by the system clock, in ms) the target stimulus was turned off

Don't modify anything in Design view unless you know what you're doing!

If you close the Design view and click **Open** instead, you see the *contents* of this table. Here is the contents of the FiveChoice_TrialData table. I entered some sample results into this table.

FiveChoice_TrialData : Table

	_RecordNum	DateTimeCode	Rat	Box	Trial	OfferedHole	ChosenHole	PrematureNosepok	Correct	Incorrect	Omission	Sc
▶	1	05/04/2006 10:54:07	subject2	0	1	1	2	0	☑	☑	☐	
	2	05/04/2006 10:54:07	subject2	0	2	4	4	0	☑	☐	☐	
	3	05/04/2006 10:54:07	subject2	0	3	0	0	0	☑	☐	☐	
	4	05/04/2006 10:54:07	subject2	0	4	4	1	0	☐	☑	☐	
	5	05/04/2006 10:54:07	subject2	0	5	-1	-1	7	☐	☐	☐	
	6	05/04/2006 10:54:07	subject2	0	6	3	3	0	☑	☐	☐	
	7	05/04/2006 10:54:07	subject2	0	7	2	-1	0	☐	☐	☐	
	8	06/04/2006 15:19:45	subject2	0	1	4	3	0	☐	☑	☐	
	9	06/04/2006 15:19:45	subject2	0	2	1	1	0	☑	☐	☐	
	10	06/04/2006 15:19:45	subject2	0	3	2	2	0	☑	☐	☐	
	11	06/04/2006 15:19:45	subject2	0	4	0	0	0	☑	☐	☐	
	12	06/04/2006 15:19:45	subject2	0	5	-1	-1	1	☐	☐	☐	
	13	06/04/2006 15:19:45	subject2	0	6	2	2	0	☑	☐	☐	
	14	06/04/2006 15:19:45	subject2	0	7	4	4	0	☑	☐	☐	
	15	06/04/2006 15:19:45	subject2	0	8	4	-1	0	☐	☐	☐	
*	(AutoNumber)								☐	☐	☐	

Feel free to explore the tables.

When you want to extract data for analysis, you may want to create **queries** to do so. (Queries are listed in the "Queries" section of the main database screen.) Queries can be created using Access's visual query design system, or using the language SQL (Structured Query Language). A little on relational database principles and SQL follows.

1.11.4 Relational databases in general

I have found the most useful way to store data is in a **relational database,** often called a relational database management system (RDBMS). A relational database stores data in **tables,** which are made up of *fields* and *records:*

A table:			*five fields:*		
	Date	Rat	NumResponses	NumStimuli	NumReinforcements
one record:	17/2/00 12:29:00	M4	56	5	1
another:	17/2/00 14:37:06	M5	437	43	8
… and so on	17/2/00 12:54:00	M4	263	26	5

The driving principle behind a relational database is this: **never duplicate data.** Let's say our rats came from two groups, Sham and Lesion. If we wanted to record this in the database, so we could analyse data by group, we could store it like this:

Table *BigData*

Date	Rat	Group	NumResponses	NumStimuli	NumReinforcements
17/2/00 12:29:00	M4	sham	56	5	1
17/2/00 14:37:06	M5	lesion	437	43	8
17/2/00 12:54:00	M4	sham	263	26	5

However, this introduces two problems. Firstly, it generates very large tables. Secondly, and more importantly, it is unclear what to do if the data is inconsistent – let's say the underlined 'sham' was changed to 'lesion' by mistake. The database would then not know whether rat M4 was in the Sham or Lesion group – there would be entries for both. The solution to both problems is to create two tables, *linked* on the smallest possible unit of information (in this example, the rat name):

Table *Responses*

Date	**Rat**	NumResponses	NumStimuli	NumReinforcements
17/2/00 12:29:00	**M4**	56	5	1

17/2/00 14:37:06	**M5**	437	43	8
17/2/00 12:54:00	**M4**	263	26	5

Table *Groups*

Rat	Group
M4	sham
M5	lesion

By using the rat name as a **key** (also known as a *foreign* key), the database can link the two tables together whenever we want to know how many responses the two groups made on average.

When we want to find out that sort of information, we **query** the database, specifying how we want to see the data. We could, for example, obtain the following (ignoring a glaring scientific error!):

Query *AverageByGroups*

Group	**N**umberOfSubjects	MeanNumResponses	MeanNumStimuli	MeanNumReinforcements
sham	2	159.5	15.5	3
lesion	1	437	43	8

Summary of database principles

So relational databases split up the data (which should be entered in well-designed tables without any duplication of information) from queries that look at the data in an infinite variety of ways.

A concrete example: Microsoft Access 97

Microsoft Access 97 is a commonly-used relational database for PCs. It isn't perfect, by a long shot, but I've found it good enough. It supports **structured query language (SQL)** for designing queries; this is a powerful quasi-English language. For example, the query shown above would be written in SQL like this:

```
SELECT group,
    count(*) as NumberOfRats,
    avg(NumResponses) as MeanNumResponses,
    avg(NumStimuli) as MeanNumStimuli,
    avg(NumReinforcements) as MeanNumReinforcements
FROM responses, groups
WHERE responses.rat = groups.rat
GROUP BY group
;
```

If you find all this a bit cryptic, Access also provides a graphical interface for designing queries.

Getting data out of a database

Given a well-designed database, you should be able to get the data out in any conceivable way. The size of this manual doesn't permit a detailed look at relational database design or queries, but there are abundant sources. If you use Microsoft Access, there's the help system, but I also recommend Viescas JL (1997), *Running Microsoft Access 97*, Microsoft Press. Beyond that there is a whole field of database design.

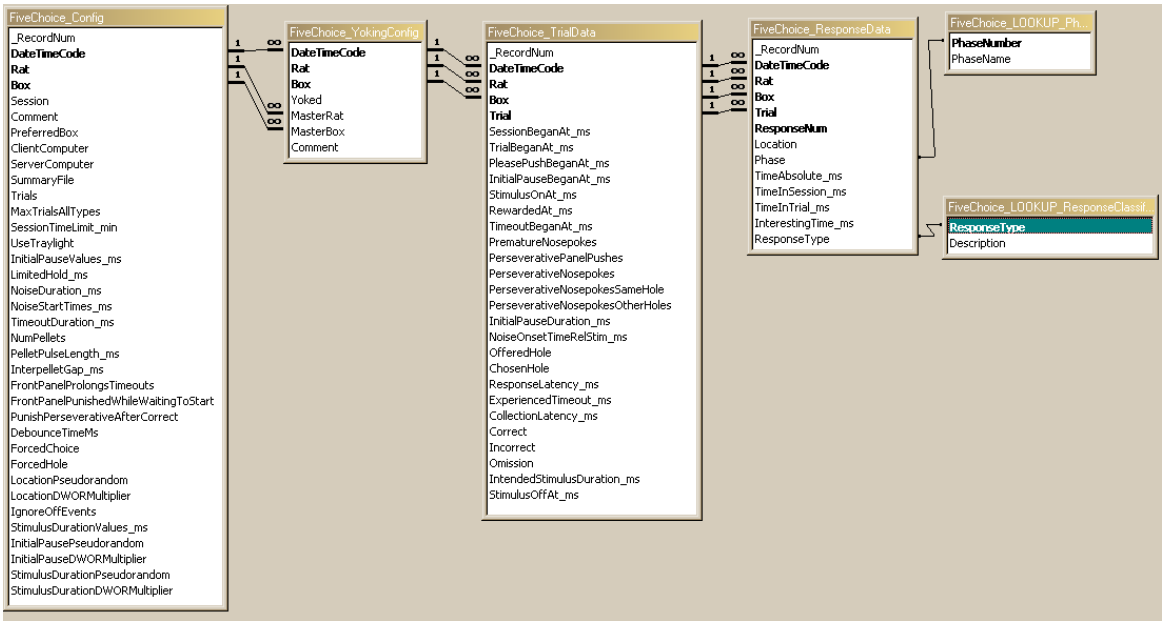
Tip

I operate on the principle that any view of the data is achievable. If the graphical query design can't do it, you can use SQL. If SQL can't do it alone, you can use Visual Basic to augment it. If all that fails (and it hasn't failed me yet) you can always re-export the data and use a general-purpose programming language to analyse it. If the data's there, you can get at it.

One thing is worth noting: modern statistical packages (e.g. SPSS, *http://www.spss.com/*) are starting to support the ODBC standard for exchanging information with databases. You can set up database queries to create views of the data that your stats packages can use, then set up sequences of ODBC capture, analysis and graphical presentation in your stats package. Then whenever you import new data, you can run the entire analysis in a matter of seconds. If you handle large volumes of data, it easily repays the initial effort.

1.11.5 Database structure

This is the structure of the FiveChoice database:



Index

- F -